UNIVERSIDAD CARLOS III

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA TELEMÁTICA

Trabajo de Fin de Grado

Aplicación Android para obtener información del estado de las carreteras.

Autor:
Elsa Gómez Martínez

Tutor:
Carlos García Rubio

Septiembre de 2016

Título: **Aplicación Android para obtener información del estado de las carreteras.**

Autor: **Elsa Gómez Martínez**

Tutor: **Carlos García Rubio**

Realizado el acto de defensa y lectura del Trabajo de Fin de Grado el día 7 de Octubre de 2016, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid (Leganés).

PRESIDENTE: David Larrabeiti López.

SECRETARIO: Rodolfo Cuerno Rejado.

VOCAL: Yoel Gustavo Yera Mora.

SUPLENTE: Pedro José Muñoz Merino.

"Solo aquellos que se atreven a tener grandes fracasos terminan consiguiendo grandes éxitos".
Robert F. Kennedy

A mis padres, por su apoyo incondicional, por enseñarme que siempre hay otra solución.

A mis hermanos, porque como todos los hermanos nos llevamos mal, pero nos queremos hasta morir.

A Julián, por ser mi héroe y mi estrella, por cuidarme desde el cielo.

A mis abuelas, por su preocupación y comprensión.

A Álvaro, por no dejarme caer, por ir de mi mano, por esperarme.

RESUMEN

El crecimiento de población en el último siglo ha conllevado un aumento de vehículos circulando por la carretera, provocando un mayor deterioro en el firme, lo cual aumenta el riesgo de que se produzcan accidentes de tráfico, así como un deterioro en el vehículo. También es un hecho que hoy en día vivimos en una época denominada "*Always On*". Cada día son más personas las que disponen de un terminal móvil que utilizan día a día en su rutina.

Los teléfonos inteligentes, también llamados *Smartphone*, forman parte de nuestra rutina. Además, gracias al gran desarrollo tecnológico, cuentan con sensores y módulos cada vez más potentes capaces de detectar aceleraciones, localización, giros, etc. Esto supone una gran ventaja pero a veces puede provocar inconvenientes.

El proyecto, "Aplicación Android para obtener el estado de las carreteras", busca crear una aplicación móvil para dispositivos Android que permite reportar con la colaboración de los usuarios donde hay desperfectos en la carretera de manera automática. Dicha aplicación utiliza los sensores para detectar la localización del desperfecto, almacenando localmente, a diferencia de otras posibles soluciones del mercado, los eventos detectados.

ABSTRACT

Population growth in the last century has involved an increase of vehicles on the road, causing a higher deterioration on the road surface, which increase the risk of traffic accidents as well as a worsening of the vehicle. It is also a fact that today we live in an epoch called “Always On”. Every day, there are more people that have a mobile terminal that they use every day in their routine.

Intelligent mobile phones also called Smartphone, are a part of our routine. Furthermore, due to the great technological development, they have sensors and modules increasingly powerful capable of detecting acceleration, location, turns, etc. This is a great advantage which can sometimes cause problems.

The project, “Android application to get the state of the roads”, seeks to create a mobile application for Android devices. This application reports automatically, with the collaboration of the users, if there is damage on the road surface. Unlike other solutions on the market, I have already explained that the application uses sensors to detect the location of the imperfections and stores the detected events locally.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS IX
ÍNDICE DE FIGURAS X
ÍNDICE DE ECUACIONES XII
ACLARACIÓN SOBRE EL TRABAJO REALIZADO ... XIII
1. INTRODUCCIÓN Y OBJETIVOS 1
1.1. INTRODUCCIÓN 1
1.2. OBJETIVOS 3
1.3. FASES DE DESARROLLO DEL PROYECTO 4
1.4. RECURSOS EMPLEADOS 5
1.5. ESTRUCTURA DE LA MEMORIA 6
2. ESTADO DEL ARTE 8
2.1. SISTEMAS OPERATIVOS 8
2.2. PLATAFORMA ANDROID 10
2.3. SENSORES Y MÓDULOS 12
2.3.1. Acelerómetro 12
2.3.2. Giroscopio 13
2.3.3. GPS 14
2.4. ESTADO DE LAS CARRETERAS 15
2.5. ENTORNO SOCIO-ECONÓMICO 17
2.6. MARCO REGULADOR 19
2.6.1. Ley Orgánica de Protección de datos de Carácter Personal 20
2.6.2. Ley General de las Telecomunicaciones 20
2.7. CONCLUSIONES SOBRE EL ESTADO DEL ARTE 21
3. ANÁLISIS Y REQUISITOS DE USUARIO 22
3.1. CASOS DE USO 22
3.2. HISTORIAS DE USUARIO 26
3.3. REQUISITOS DE RESTRICCIÓN 29
4. DISEÑO Y DESARROLLO DE LA APLICACIÓN 31
4.1. ASPECTOS TEÓRICOS 31
4.1.1. Vehículo en reposo 33
4.1.2. Vehículo en movimiento 33
4.1.3. Cálculo de la matriz de rotación y transformación de valores 34
4.2. ARQUITECTURA DEL SISTEMA 35
4.3. MÓDULOS DESARROLLADOS 37
4.3.1. Módulo de localización 38
4.3.2. Módulo de detección de movimientos 39
4.3.3. Módulo de reorientación 40
4.3.4. Módulo de almacenamientos de datos 42

4.3.5. Clases de los módulos 43
4.4. DESARROLLO DE LA APLICACIÓN FINAL 45
4.4.1. Obtención y análisis de datos 45
4.4.2. Aplicación final 49
4.4.3. Interfaz gráfica 51
4.4.4. Diagramas de flujo 53
5. ENSAYOS Y ANÁLISIS DE LOS RESULTADOS 58
5.1. ENSAYOS PRIVADOS 58
5.1.1. Ámbito de pruebas privadas 58
5.1.2. Pruebas básicas de usuario 60
5.2. ENSAYOS PÚBLICOS 62
5.2.1. Cuestionario y resultados 62
5.2.2. Análisis de las muestras 64
6. GESTIÓN DEL PROYECTO 68
6.1. ETAPAS DEL PROYECTO 68
6.2. PRESUPUESTO 73
6.2.1. Coste de personal 73
6.2.2. Coste de material 74
6.2.3. Costes indirectos 75
6.2.4. Coste total 76
7. CONCLUSIONES Y FUTURAS MEJORAS 77
7.1. CONCLUSIONES 77
7.2. FUTURAS MEJORAS 77
ANEXO A – MANUAL DE USUARIO 79
1. CALIBRAR EL DISPOSITIVO 81
1.1. Fase de calibración en reposo 82
1.2. FASE DE CALIBRACIÓN EN MOVIMIENTO: 82
2. OBTENCIÓN DE DATOS 83
2.1. Toma de datos manual 84
2.2. Toma de datos automática 84
3. CARGAR/GUARDAR CALIBRADO EL DISPOSITIVO 84
3.1. Guardar un calibrado 84
3.2. Cargar un calibrado 85
ANEXO B – GLOSARIO 86
ENGLISH VERSION 87
BIBLIOGRAFÍA 104

ÍNDICE DE TABLAS

Tabla 2.1. Resumen de las características de los SO Android e iOS. [..... 9
Tabla 3.1. Plantilla de los casos de uso. 23
Tabla 3.2. CU-01. 23
Tabla 3.3. CU-02. 23
Tabla 3.4. CU-03. 24
Tabla 3.5. CU-04. 24
Tabla 3.6. CU-05. 24
Tabla 3.7. CU-06. 25
Tabla 3.8. CU-07. 25
Tabla 3.9. CU-08. 25
Tabla 3.10. CU-09. 26
Tabla 3.11. Plantilla de las historias de usuario. 26
Tabla 3.12. Plantilla de las tareas. 26
Tabla 3.13. Historias de usuario. 27
Tabla 3.14. Tareas HU - 01. 27
Tabla 3.15. Tareas HU - 02. 28
Tabla 3.16. Tareas HU - 03. 28
Tabla 3.17. Tareas HU - 04. 29
Tabla 3.18. Plantilla de los requisitos de restricción. 29
Tabla 3.19. RR – 01. 30
Tabla 3.20. RR – 02. 30
Tabla 3.21. RR – 03. 30
Tabla 3.22. RR – 04. 30
Tabla 3.23. RR – 05. 30
Tabla 3.24. RR – 06. 30
Tabla 4.1. Métodos del módulo de reorientación. 44
Tabla 4.2. Métodos del módulo de almacenamiento. 45
Tabla 4.3. Elementos del diagrama UML. 54
Tabla 5.1. Plantilla de pruebas. 60
Tabla 5.2. Tabla de Test de Software. 61
Tabla 5.3. Muestra de respuestas al cuestionario. 64
Tabla 6.1. Estimación inicial del proyecto en horas. 68
Tabla 6.2. Duración final del proyecto en horas. 70
Tabla 6.3. Coste del personal. 74
Tabla 6.4. Coste del material. Hardware y software. 75
Tabla 6.5. Otros costes. 75
Tabla 6.6. Coste total. 76

ÍNDICE DE FIGURAS

Figura 1.1. Estado del firme de las carreteras en España, 2016. 2
Figura 2.1. Cuota de mercado de los sistemas operativos móviles, 2016. 8
Figura 2.2. Ciclo de vida de un *Activity*. 11
Figura 2.3. Pila de software de Android. 12
Figura 2.4. Acelerómetro. 13
Figura 2.5. Giroscopio. 14
Figura 2.6. Funcionamiento del GPS. 15
Figura 2.7. Evolución del estado de los firmes. 16
Figura 2.8. Logo de la aplicación *Street Bump*. 17
Figura 2.9. Aplicación *Street Bump*. 18
Figura 2.10. Logo de la aplicación *Avisos Madrid*. 18
Figura 2.11. Aplicación *Avisos Madrid*. 19
Figura 3.1. Diagrama de los casos de uso. 22
Figura 4.1. Ejes cartesianos del coche. 32
Figura 4.2. Arquitectura de diseño del sistema. 37
Figura 4.3. Interfaz de la aplicación GPS. 38
Figura 4.4. Interfaz de la aplicación acelerómetro. 39
Figura 4.5. Diagrama de flujo del módulo de reorientación. 41
Figura 4.6. Interfaz gráfica de la aplicación de reorientación. 42
Figura 4.7. Interfaz gráfica de la aplicación de almacenamiento de datos. 43
Figura 4.8. Interfaz de la aplicación de recopilación de datos. 46
Figura 4.9. Formato inicial del almacenamiento de datos. 47
Figura 4.10. Formato final del almacenamiento de datos. 47
Figura 4.11. Gráfico de los datos de baches y badenes obtenidos manualmente. 48
Figura 4.12. Definición gráfica de bache. 49
Figura 4.13. Definición gráfica de badén. 49
Figura 4.14. Gráfico de los datos obtenidos manualmente y automáticamente, 12 de Julio. 50
Figura 4.15. Gráfico de los datos obtenidos manualmente y automáticamente, 15 de Julio. Segunda acotación. 51
Figura 4.16. Interfaz de usuario principal. 52
Figura 4.17. Interfaz de usuario con menú desplegable. 53
Figura 4.18. Diagrama del flujo principal. 55
Figura 4.19. Diagrama de flujo del botón "Bache". 56
Figura 4.20. Diagrama de flujo de los botones "Resalto" y "Calibrar". 56
Figura 4.21. Diagrama de flujo de la detección automática de baches. 57
Figura 5.1. Entornos iniciales de pruebas. 58
Figura 5.2. Entorno de pruebas. Carretera M - 40. 59
Figura 5.3. Entorno de pruebas. Carretera de ciudad. 59
Figura 5.4. Cuestionario para los usuarios. 62
Figura 5.5. Gráficos de sectores de las respuestas obtenidas. 63
Figura 5.6. Representación de muestra del primer usuario. 65
Figura 5.7. Representación de la muestra del segundo usuario. 66
Figura 5.8. Representación de la muestra del tercer usuario. 67
Figura 6.1. Diagrama de Gantt de la estimación inicial. 69
Figura 6.2. Diagrama de Gantt de la estimación final. 72
Figura 6.3. Salarios de ingenieros junior y senior usando *ActiBVA*. 73

Figura A.1. Pantalla principal y menú desplegable..... 79
Figura A.2. Solicitud de permisos al usuario. 80
Figura A.3. Vista principal de la aplicación. 81
Figura A.4. Fase en reposo. 82
Figura A.5. Fase en movimiento..... 83
Figura A.6. Guardar calibrado. 84
Figura A.7. Cargar calibrado. 85
Figure C.1. Pavement Surface condition of the roads in Spain. 89

ÍNDICE DE ECUACIONES

Ecuación 4.1. Ecuación del ángulo de inclinación. 33
Ecuación 4.2. Ecuación del ángulo de pre-rotación. 33
Ecuación 4.3. Ecuación del ángulo de post-rotación. 34
Ecuación 4.4. Ecuación de la matriz de rotación. 34
Ecuación 4.5. Ecuación de $R\psi$, $R\theta$ y $R\phi$. 34
Ecuación 4.6. Cálculo de los valores para el sensor reorientado. 35
Ecuación 6.1. Ecuación del coste imputable. 74

ACLARACIÓN SOBRE EL TRABAJO REALIZADO

Antes de iniciar la lectura de este Trabajo de Fin de Grado es importante realizar una serie de aclaraciones:

Este proyecto comparte su núcleo con el trabajo *"Aplicación Android para obtener información de tráfico en carreteras"*, cuyo autor es Álvaro González Caballero [39]. La idea es crear aplicaciones que utilicen los diferentes sensores de un Smartphone, para captar eventos e interaccionar con el entorno, ante distintos problemas planteados.

Para agilizar el desarrollo, y puesto que al iniciar el proyecto ambos desconocíamos el entorno de programación Android, se decide realizar una serie de módulos de forma conjunta, que sirvan como base para posteriormente realizar nuestras soluciones independientes. Esto no sólo ayuda en el aprendizaje, sino que también ayuda en la investigación y establecimiento de una filosofía común.

Los módulos arriba mencionados, serán explicados durante el transcurso de esta memoria. Para indicar que estos han sido desarrollados de forma conjunta, durante la documentación haré anotaciones aclaratorias en cursiva. No obstante, dejo una lista de los módulos a continuación:

- Módulo de localización (Aprendizaje del uso GPS)
- Módulo de detección de movimientos (Aprendizaje del uso del acelerómetro)
- Módulo de almacenamiento de datos (Aprendizaje del uso de ficheros en Android)
- Módulo de reorientación (Aplicación de los conocimientos adquiridos mediante el artículo *TrafficSense* [12]).

1. INTRODUCCIÓN Y OBJETIVOS

En este capítulo se realiza una breve introducción al proyecto realizado, así como la motivación por la cual se ha llevado a cabo y los objetivos a conseguir. Además, se comentarán las fases de desarrollo del proyecto, los recursos empleados y la estructura del documento.

1.1. Introducción

El mundo de la telefonía ha avanzado enormemente desde el primer teléfono inteligente (en adelante *Smartphone)* en 1994 [1] de tal manera que hemos pasado de usarlo para funciones limitadas a integrarlo en nuestra rutina, lo que ha permitido que se desarrollen Smartphone con mayor capacidad y rendimiento. Esto, unido a las diferentes plataformas que han ido apareciendo con el tiempo, ha permitido la creación de un ilimitado número de aplicaciones móviles que cubren infinidad de necesidades que un usuario de este gadget pueda tener.

También es un hecho que el uso del automóvil aumenta año tras año. Estudios de la Intensidad Media Diaria (IMD) de la Comunidad de Madrid así lo demuestran. [2] El año pasado el tráfico aumentó un 2,64% respecto a los estudios de 2014 en la Comunidad de Madrid. Dicho estudio hace mención a su vez al parque de vehículos de la Comunidad de Madrid, que aumentó un 0,19% contando con un parque total de 4.200.832 vehículos.

Estos dos elementos, telefonía y vehículo son puntos clave en el desarrollo del proyecto orientado a un usuario de ambos.

La Asociación Española de la Carretera (AEC) destaca que el estado de conservación de las vías requiere una inversión de 6.617 millones de euros para devolverla a una situación "adecuada". El estudio realizado por la AEC analiza 3.000 tramos de la red viaria española entre los cuales sólo se puede considerar el firme en estado aceptable en el País Vasco y Extremadura. [3]

Según las estadísticas de la DGT (Dirección General de Tráfico), un 10% de los accidentes de tráfico es debido al mal estado de las carreteras [4] [5]. Por otro lado, la Fundación CEA (Comisariado Europeo del Automóvil) ha tachado de insuficiente la inversión destinada a la conservación de carreteras españolas. Sólo un 33% de las carreteras estatales y un 45% de las carreteras autonómicas pueden considerarse como “buenas". Los datos de Automovilistas Europeos Asociados (AEA) explican que esto es debido a que el Ministerio de Fomento redujo el presupuesto destinado a la conservación de las carreteras en un 40% entre 2008 y 2012. [6]

El mal estado del firme causa accidentes de tráfico, en ocasiones mortales, además de deterioros en los automóviles (ruedas, amortiguadores, elementos de la dirección, etc.).

Figura 1.1. Estado del firme de las carreteras en España, 2016. [3]

Las diversas soluciones existentes son consideradas poco eficientes o excesivamente caras. Por un lado, la DGT, con vehículos de mantenimiento, reporta el estado de señales o el estado del firme de manera manual, lo que se convierte en un método poco eficiente e inexacto.

Por otro lado, los fabricantes de vehículos *Land Rover* y *Jaguar* han desarrollado un sistema llamado *MagneRide*, que permite detectar el estado de la carretera de manera automática ya que constan de sensores especializados. Estos sensores han de ser muy precisos, lo que incrementa su coste y hacen que esta solución se convierta en algo inalcanzable económicamente hablando[7]. También nos encontramos con una iniciativa por parte de Bose que pretendía mantener los movimientos de la carrocería cercanos a cero. Finalmente fue desechada ya que los fabricantes de automóviles la consideraron pesada y cara [8].

Conjuntamente, podemos encontrar campañas de prevención como la llevada a cabo por Ponle Freno [9], la Fundación CEA [10] o la AEA [11] que piden denunciar esta situación localizando grietas, baches y socavones para reducir el número de accidentes en carretera.

Este proyecto trata de localizar los baches del firme y así reducir el número de accidentes provocados por esta situación. Dado el avance de la tecnología durante los últimos años, el aumento del uso de los Smartphone, y el incremento de automóviles circulando, se ha decidido crear una aplicación capaz de detectar automáticamente un bache, y guardar su localización usando sensores del propio dispositivo móvil como el GPS y el acelerómetro.

En la actualidad, existen diferentes sistemas operativos móviles pero destacan dos principalmente: el sistema operativo Android (Android SO, en adelante) e iOS. La plataforma elegida para el desarrollo de este proyecto es Android ya que posibilita el acceso a un mayor número de usuarios entre otras ventajas que se analizarán en el capítulo 2.

1.2. Objetivos

El Trabajo de Fin de Grado desarrollado detecta irregularidades en las carreteras, como pueden ser baches. La solución deberá ser alternativa a las actuales, permitiendo el desarrollo de una solución barata, fácil de usar, que proteja al usuario y evite los daños personales al mismo (en posibles accidentes causados por baches). A continuación, explicamos estos objetivos más detalladamente:

- **Ahorro en costes:** Este proyecto deberá desarrollar una solución barata, que pueda ser utilizada por el público general. Por esta razón se utilizan los dispositivos móviles y la plataforma Android.

- **Facilidad en el uso:** La solución deberá ser de manejo sencillo, por lo que se implementará un mecanismo automático para detectar baches.

- **Protección al usuario:** El respeto por la privacidad del usuario, y el ahorro de los recursos de su dispositivo (como la batería), será clave para conseguir que la aplicación se expanda y cuente con numerosas fuentes de datos.

- **Reducción de los accidentes:** De conseguirse los objetivos anteriores, podría verse disminuido el número de accidentes como vimos en estadísticas anteriores. Con la colaboración de todos se pueden salvar vidas.

Por tanto, se puede resumir el objetivo principal en: desarrollar una aplicación abierta al público general (por su bajo coste, y necesidad mínima de recursos), que permita detectar baches y por consiguiente evitar accidentes (mejorando el estado de las carreteras).

1.3. Fases de desarrollo del proyecto

Para un mayor entendimiento y estructuración, este proyecto se ha dividido en diferentes fases que se detallan a continuación. Es importante aclarar que *las fases de documentación, formación y parte de la fase de implementación, se han llevado a cabo de forma común con el trabajo de fin de grado "Aplicación Android para obtener información de tráfico en carreteras", cuyo autor es Álvaro González Caballero.* [39] (aclaración al lector)

Fase 1 – Documentación: Durante la fase de documentación se procede a examinar el problema, buscando la mejor manera de detectar el mal estado del firme. Se estudian los requisitos que ha de seguir la aplicación tales como necesidad de un calibrado previo en el Smartphone, que el terminal disponga de los sensores que requiere la aplicación, etc. y se buscan soluciones alternativas que cumplan con los objetivos detallados en el apartado anterior. Esta primera fase nos permitirá abordar con mayor facilidad las fases posteriores ya que tendremos una idea clara del problema a tratar.

Algunos de los documentos consultados durante la fase de documentación se nombran a continuación:

- *TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones*. Abril 2008. Prashanth Mohan, Venkata N. Padmanabhan and Ramachandran Ramjee. [12]
- *Computing Euler angles from a rotation matrix*. *Gregory G.* Slabaugh. [13]
- *Using the Rotation Vector Sensor*. API Android. [14]
- *Dynamics of Tree-Type Robotic Systems,* Springer, 2013. Suril Vijaykumar Shah, Subir Kumar Saha, Jayanta Kumar Dutt. [15]

Fase 2 – Formación: En esta fase se decide analizar la plataforma que concierne al proyecto, Android, estudiando las características de la misma. Con el software Android Studio, considerado el IDE (*Integrated Development Environment*) oficial para el desarrollo de aplicaciones Android, junto con los conocimientos adquiridos durante toda la carrera en lenguajes de programación como Java, podremos comprender los aspectos teóricos de Android.

Los recursos utilizados para la formación en Android han sido:

- *Curso de Android de la asignatura Aplicaciones móviles*. Celeste Campo Vázquez y Carlos García Rubio, 2015 [16].

- *Curso Programación Android.* Salvador Gómez Oliver, Noviembre 2011 [17].
- API Android [14].
- *Android 100%*. Ramón Invarato Menéndez, Octubre 2014 [18].

Fase 3 – Implementación: Con la formación adquirida en la segunda fase, se procede a desarrollar la aplicación. Para ello se crean los casos de uso de la aplicación y se exponen los requisitos. Además, se crean diferentes módulos, que ayudarán a conocer el uso de los sensores del Smartphone, así como de los ficheros que la aplicación requiere. Conociendo su funcionamiento, se procede a realizar un modelo en fase de pruebas de la aplicación.

Fase 4 – Pruebas y corrección de errores: En la Fase 4 se realizan pruebas, públicas y privadas, sobre el modelo de la aplicación creado en la Fase 3 y se procede a solventar errores existentes o mejorar el funcionamiento de la aplicación. Las pruebas realizadas se detallarán en el capítulo 5.

Fase 5 – Análisis de datos con MATLAB: Una vez realizadas las pruebas y solucionados los problemas, se examinan los datos obtenidos durante las mismas a través del software MATLAB.

Fase 6 – Aplicación final y Documentación: Durante esta fase se da por finalizada la aplicación y se procede a realizar una versión final del documento actual teniendo en consideración las notas tomadas durante las fases previas.

1.4. Recursos empleados

Durante la realización del proyecto, se han empleado recursos hardware, software y vehículos. En esta sección explicaré de forma general cuales han sido esos recursos y su utilidad (para una descripción exhaustiva de los recursos ver el capítulo 6)

Smartphone: Se han utilizado diferentes Smartphone, que han permitido probar la aplicación de forma real. En todos los Smartphone probados el requisito era contar con un software de Android 4.2 (*Jelly Bean*) o superior.

Ordenador: Se han utilizado diversos ordenadores portátiles. Su uso principal ha permitido el desarrollo de la aplicación, y la documentación en esta memoria. El SO de estos ha sido Windows y MacOS. El software utilizado en ellos es Android Studio, Matlab y el paquete Office.

Vehículo: Para la realización de pruebas en las carreteras, se han utilizado diferentes modelos de vehículo. Para tener una gran muestra, se han usado vehículos de diferentes segmentos y antigüedad.

1.5. Estructura de la memoria

En este apartado se realiza una descripción general de la estructura del documento.

Introducción y objetivos: En este primer capítulo se realiza una breve introducción al proyecto realizado: localización de baches en el firme, así como los motivos para llevarlo a cabo. Para ello se describirán los objetivos a conseguir, las fases de desarrollo del proyecto, los recursos empleados y, por último, la estructura del documento.

Estado del arte: En este capítulo se analizará el contexto tecnológico del TFG (Trabajo de Fin de Grado). Para ello, se realiza un análisis de los medios utilizados (dispositivos, tecnologías y firmes), así como las soluciones que existen actualmente y el marco regulador. Al final de este capítulo se tendrán los conocimientos necesarios para entender las soluciones presentadas en los capítulos posteriores, conociendo las limitaciones que las tecnologías utilizadas presentan.

Análisis y requisitos de usuario: Con las conclusiones y conocimientos adquiridos del estado del arte, podremos analizar los diferentes requisitos del proyecto. Para ello se realizará un análisis de los casos de uso, las historias de usuario y los requisitos de restricción del proyecto. Este apartado nos ayudará a identificar las necesidades del proyecto. En cada una de las secciones se explicará el porqué de la elección de esta metodología.

Diseño y desarrollo de la aplicación: A lo largo del capítulo se analizarán los aspectos generales del sistema, explicando los inicios de la aplicación, su desarrollo y la investigación realizada. Asimismo, se explicarán las bases teóricas que dan lugar a la implementación, se mostrarán diagramas de flujo que ayuden a entender la aplicación y se hablará de la logística interna.

Ensayos y análisis de los resultados: En este capítulo se expondrán los ensayos realizados como método de prueba de la aplicación, y el análisis de los resultados obtenidos. La división de las secciones en ensayos privados y públicos, tiene la intención de mostrar los diferentes ensayos efectuados.

Gestión del proyecto: En este capítulo se expondrá la gestión del proyecto, de esta forma se podrán planificar y orientar los procesos seguidos en este TFG de manera metódica. Para ello se analizarán las etapas del mismo, añadiendo el diagrama de Gantt para explicar el tiempo previsto para las diferentes tareas. Además, se definirán los costes personales, los costes de material y los costes indirectos detallando cada uno de ellos.

Conclusiones y futuras mejoras: En este capítulo se detallarán las conclusiones obtenidas con la realización de este proyecto. Además, se expondrán líneas de trabajo futuras que mejorarán la aplicación.

Anexo A – Manual de Usuario: Este anexo sirve como breve guía para el usuario. En él se explican los pasos a seguir para utilizar la aplicación.

Anexo B - Glosario: En este capítulo se definen los acrónimos utilizados a lo largo del proyecto.

English versión: Según la normativa aplicada a los alumnos del plan Bolonia (plan 2011), es necesario la presentación de las siguientes partes en lengua inglesa:

- Introducción completa en lengua inglesa.
- Resumen del proyecto 5 a 10 hojas en lengua inglesa. Desde el capítulo 2 hasta el capítulo 6 inclusive.
- Conclusiones completas en lengua inglesa.

Bibliografía: Exposición de las fuentes consultadas para la realización del proyecto.

2. ESTADO DEL ARTE

Este capítulo tiene como objetivo analizar el contexto tecnológico en el que se desarrolla este proyecto. Se realiza un análisis del estado del arte de los medios utilizados, así como las soluciones que existen actualmente y el marco regulador de éstas.

Al final de este capítulo se tendrán los conocimientos necesarios para entender las soluciones presentadas en los capítulos posteriores, conociendo las limitaciones que las tecnologías utilizadas presentan.

2.1. Sistemas operativos

Un sistema operativo móvil (en adelante SO móvil) es un sistema capaz de controlar un dispositivo móvil, lo que se traduce en una interpretación de lo que el usuario quiere que el terminal haga. Además, a diferencia de un SO, el SO móvil está diseñado para ser más rápido y estar dirigido a la conectividad inalámbrica. [19]

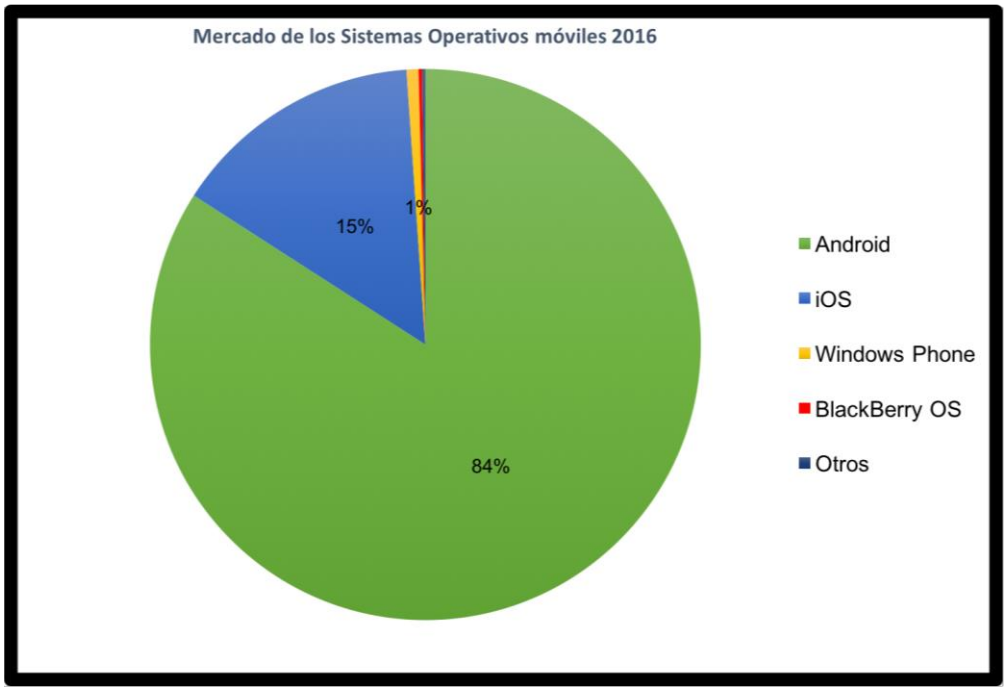


Figura 2.1. Cuota de mercado de los sistemas operativos móviles, 2016. [19]

En la Figura 2.1 observamos que existen diferentes sistemas operativos móviles tales como Android SO, iOS, Windows Phone, BlackBerry OS etc. Actualmente destacan dos más principales: Android SO, creado por Google e iOS, de la marca Apple.

A pesar de haber optado de antemano por el SO Android, merece analizar brevemente las características de los dos sistemas operativos con más valor en el mercado y así secundar mi elección.

Android SO: Una de las principales características que definen a Android es ser un sistema operativo de código abierto. Gracias a ello, cualquier fabricante puede implementar este SO en su propio hardware, por lo que existen Smartphone de diferentes gamas y precios que cuentan con Android. Esto sin duda es un beneficio para los desarrolladores de aplicaciones, ya que incrementa el número de clientes potenciales para su aplicación. Asimismo, un SO Android da la posibilidad de descargar aplicaciones de tiendas diferentes a *Google Play*. Además, goza de una interfaz capaz de adaptarse a las preferencias del usuario, dando lugar a un Smartphone muy personalizado.

En cuanto a las actualizaciones de este sistema operativo móvil, no siempre podemos disponer de la última versión ya que depende de varios factores, pero siempre la podemos obtener de manera extra-oficial (a través de *RO*Ms, memorias de solo lectura).

En lo referente al desarrollo de aplicacines contamos con el lenguaje bien conocido, Java, y además con un sistema de testeo más real debido a su unidad central de procesamiento (CPU) virtual.

iOS: Es un sistema operativo de código cerrado, únicamente orientado a dispositivos de su propia marca, Apple. Esto le confiere algunas ventajas, como la gran seguridad ante ataques externos que tienen estos dispositivos. Esta seguridad es debida al gran control al que Apple somete a sus aplicaciones antes de ser publicadas en su APP Store. Otra de sus ventajas es la posibilidad de sincronizar nuestro Smartphone con otros dispositivos de la marca.

iOS permite actualizar a su última versión de software tanto a dispositivos nuevos como a los antiguos, siempre y cuando estén en el mercado. A pesar de ello, cuenta con una interfaz predeterminada y sus aplicaciones sólo están disponible a través de APP Store (tienda de aplicaciones).

Para el desarrollo de aplicaciones móviles en este SO móvil necesitamos tener conocimientos en lenguajes de programación tales como *Swift* u *Objective-C*. Además, su simulador de aplicaciones, aunque es más veloz, es menos realista.

	Android SO	iOS
Código	Abierto	Cerrado
Precios	Gama abierta	Prefijados
Interfaz	Personalizada	Predeterminada
Descarga Aplicaciones	Google Play y otros	APP Store
Actualizaciones	Sujeto a condiciones	Última versión
Lenguaje de desarrollo	Java	Objective-C/Swift
Testeo aplicaciones	Real	Rapidez

Tabla 2.1. Resumen de las características de los SO Android e iOS. [20]

En base a las características analizadas anteriormente, así como basándonos en el 84% de usuarios que disponen de un terminal con SO Android y unido a su código abierto, su lenguaje de desarrollo de aplicaciones y su coste menor, Android es la plataforma elegida para el desarrollo de este proyecto.

2.2. Plataforma Android

Android es un sistema operativo en sus inicios pensado para teléfonos móviles, y actualmente para tablets, relojes inteligentes (también llamados smartwatches), televisiones, e incluso automóviles, personalizable y fácil de usar, desarrollado por Google.

Lo que diferencia a Android de otros sistemas operativos móviles es su núcleo Linux, un núcleo libre, gratuito y multiplataforma. Esto, unido a la facilidad que proporciona a los desarrolladores de aplicaciones móviles al no tener que preocuparse por el hardware, lo convierte en una de las mejores soluciones para el desarrollo de software móvil.

Este sistema proporciona todas las interfaces necesarias para realizar aplicaciones que accedan a todas las funciones del teléfono tales como: GPS, cámara, almacenamiento, etc. También, cuenta con la ventaja de permitir programar en el lenguaje Java, además de que sus herramientas son gratuitas. [14]

La arquitectura de componentes con la que cuenta Android es la siguiente:

- **Kernel:** El kernel o núcleo Linux, es la primera capa de la pila de software de Android cuya función es hacer de intermediario entre el hardware y el resto de capas de la pila. Además, gestiona los servicios base del sistema tales como recursos de memoria, seguridad, drivers, etc.
- **Android Runtime:** Es el entorno de ejecución de aplicaciones utilizado por este SO móvil. Android Runtime (ART) sustituyó a *Dalvik*, la máquina virtual que utilizaba Android hasta la versión 5.0. Su diferencia con *Dalvik* la encontramos en la compilación de las aplicaciones, ART compila totalmente en el momento de la instalación mientras que *Dalvik* compila el código cada vez que se inicia la aplicación.
- **Bibliotecas/Librerías:** Situadas justo encima del kernel. Estas librerías son nativas de C/C++ y son utilizadas por varios componentes del sistema. Son instaladas en el terminal por el fabricante antes de ponerlo a la venta. Algunos ejemplos de ellas son las bibliotecas multimedia (audio, imagen y video), motores de navegación, etc. Su objetivo es proporcionar funcionalidad a las aplicaciones para que tareas que se repiten con mucha frecuencia se lleven a cabo de la manera más eficiente.

- **Framework de aplicaciones:** El *framework* de aplicaciones es un patrón utilizado para el desarrollo o implementación de aplicaciones. Proporciona el acceso al API (*Application Programming Interface*) de Java para rehusar sus funcionalidades o modificarlas, lo que facilita crear aplicaciones en base a un esqueleto ya creado.
  Un ejemplo de algunos de los elementos que forman parte de esta capa:

  - *Activity Manager*: Cada aplicación consta de un *Activity* que permite la iteración por parte del usuario con la APP. Además, tienen un ciclo de vida como podemos apreciar en la Figura 2.2. Si cambiamos de aplicación mientras estamos ejecutando otra, cambiaremos de *Activity*. Por lo tanto, el *Activity Manager* administra la pila de actividades y su ciclo de vida.

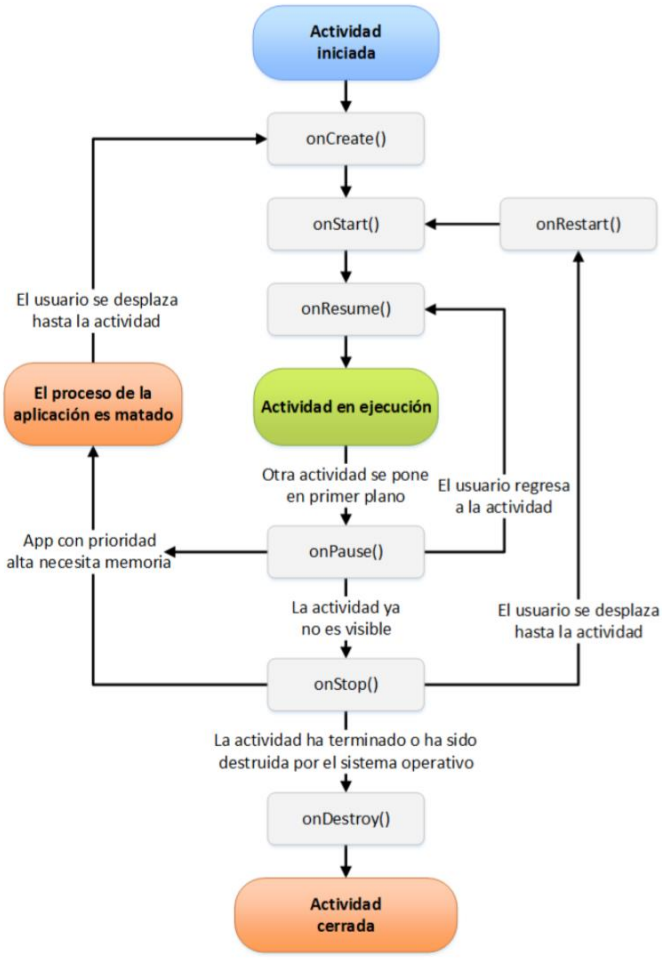


Figura 2.2. Ciclo de vida de un *Activity*. [18]

- *Content Provider*: Los *Content Providers*, son un mecanismo creado por Android para poder compartir datos entre aplicaciones.

- **Aplicaciones:** La última capa de la pila de software son las aplicaciones. Cualquier aplicación programada en Java pertenece a esta capa. Android proporciona algunas de ellas por defecto tales como contactos, correo electrónico, navegador, etc.

Figura 2.3. Pila de software de Android. [21]

2.3. Sensores y módulos

Los dispositivos móviles actuales contienen sensores de todo tipo. Para simplificar tareas cotidianas, estos recogen una gran cantidad de datos. También llevan instalados diferentes módulos tales como: el módulo WI-FI o el GPS. Aun así, dependiendo de la gama del teléfono podemos encontrarnos con diferentes tipos de sensores.

A pesar de que en la actualidad los terminales poseen al menos 10 tipos entre sensores y módulos, explicaremos en ligero detalle los que afectan a este proyecto.

2.3.1. Acelerómetro

El sensor acelerómetro mide la aceleración y las fuerzas provocadas por la gravedad, con el fin de detectar el movimiento y la orientación del terminal, es decir, si nuestro terminal se encuentra en horizontal o en

vertical. Puede ser de varios tipos (mecánico, capacitivo, óptico, etc.), y convierte en una señal eléctrica estímulos físicos tales como aceleraciones, cambios de velocidad y giros. [22]

La función del acelerómetro, como su nombre indica, es medir la aceleración con la que desplazamos linealmente el teléfono, con el fin de determinar si el teléfono ha sido desplazado en cualquier dirección.

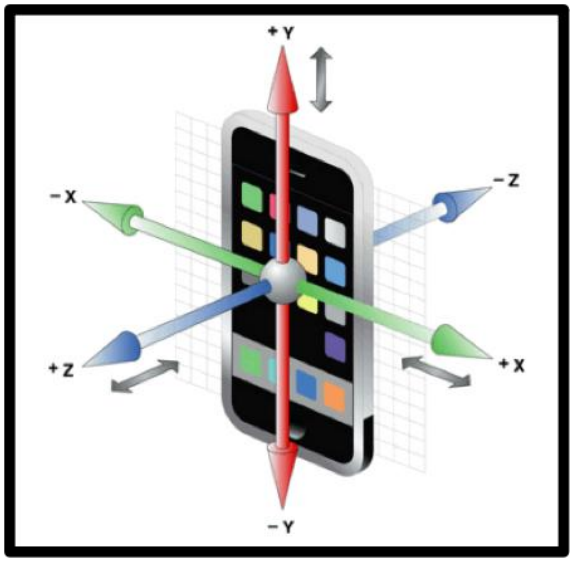


Figura 2.4. Acelerómetro. [22]

El acelerómetro es un chip de silicio que dispone de tres pequeños tubos que simulan los ejes de coordenadas tridimensionales (X, Y, Z), además de componentes electrónicos como por ejemplo condensadores.

En lo referente a sus aplicaciones (APPs), podemos destacar las APPs de actividad física, dado que son actividades que se realizan en movimiento, el acelerómetro se encarga de medir la distancia recorrida y la velocidad a la que te encuentras para posteriormente realizar estadísticas y cálculos con estas medidas. El acelerómetro es muy útil también como dispositivo de seguridad, este se encuentra en el airbag de los coches ya que, al frenar de forma brusca, el acelerómetro detecta una aceleración muy fuerte y hace que el airbag se despliegue. [23]

2.3.2. Giroscopio

El giroscopio trabaja conjuntamente con el acelerómetro, aunque su función es diferente, se encarga de medir el giro de un dispositivo en dirección diagonal, gracias a la aceleración angular, permitiendo cambiar su orientación. Por ello, entre el acelerómetro y el giroscopio, podemos detectar cambios del dispositivo en 6 ejes. [24]

El giroscopio está formado por un cuerpo con simetría de rotación que gira alrededor de un eje de dicha simetría. Aunque su funcionamiento es

complejo, básicamente gira sobre su propio eje y mantiene su orientación a pesar de que se le aplique un movimiento con diferente sentido.

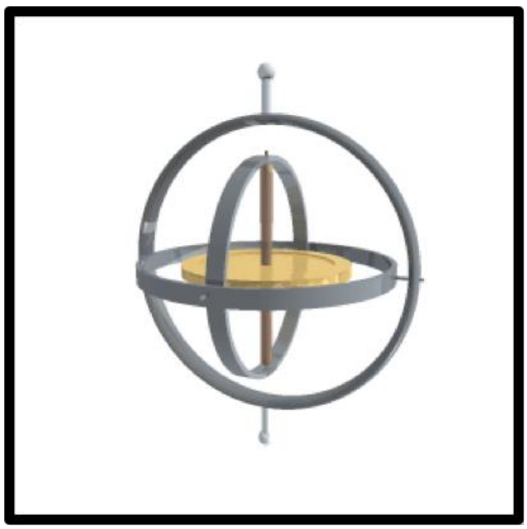


Figura 2.5. Giroscopio. [25]

La realidad virtual y el visionado de esta, son aplicaciones clave para comprender la función del giroscopio. Gracias a él podemos adentrarnos en el mundo virtual mientras nuestro dispositivo controla como cuadrar la imagen con la posición y el movimiento del jugador.

2.3.3. GPS

Hoy en día todos los Smartphone cuentan con un módulo GPS que, utilizando la red de satélites del sistema de posicionamiento global, permite hacer uso de las funciones de geolocalización y conexión.

Cuando nuestro Smartphone quiera localizarnos, se conectará a la red GPS (que consta de 24 satélites en órbita, 32 contando los satélites que mejoran la precisión y que cubren toda la superficie de nuestro planeta) tratando de conectar con el mayor número de satélites posible para averiguar nuestra posición. [26]

Para determinar la posición, básicamente, el módulo localiza cuatro o más satélites, calcula la distancia que le separa de cada uno de ellos y obtiene la posición en base a esta información. Cada vez que se enciende el GPS, se inicia un proceso de cálculo que consiste en los siguientes pasos:

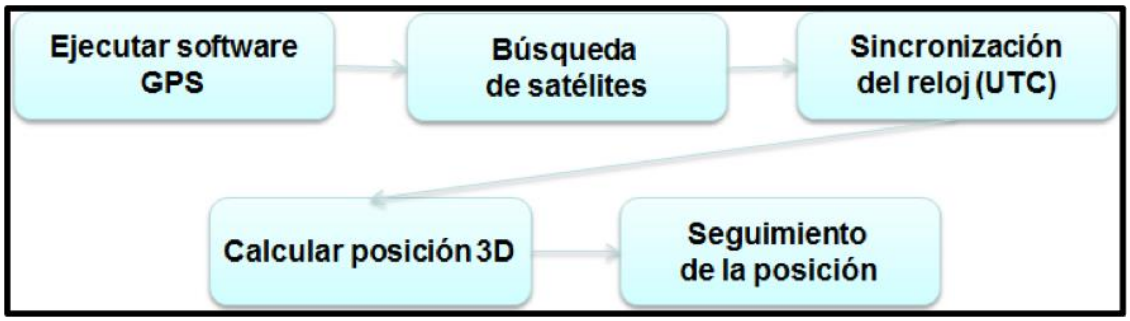


Figura 2.6. Funcionamiento del GPS. [27]

- **Ejecutar software GPS:** Se ejecuta el software interno capaz de calcular la posición.
- **Búsqueda de satélites:** Localizar el mayor número de satélites posibles. Además, es necesario recibir una señal de potencia suficiente para considerar la fuente fiable.
- **Sincronización del reloj (UTC):** Conocer la distancia a la que se encuentran los satélites de nosotros y su posición para obtener nuestra posición con cálculos matemáticos. Debido a la gran distancia a la que están los satélites es necesario ajustar nuestro reloj al reloj atómico del satélite.
- **Calcular la posición 3D:** Calculamos la posición mediante cálculos geométricos una vez conocemos la distancia y tenemos los relojes sincronizados.
- **Seguimiento de la posición:** El GPS sigue recibiendo información de los satélites y actualizando la posición.

Existen multitud de aplicaciones que requieren el uso del GPS de nuestro Smartphone, como, por ejemplo, compartir nuestra ubicación en redes sociales, que ciertas aplicaciones nos sigan y nos informen sobre lo que hay a nuestro alrededor, o la bien conocida Google Maps.

2.4. Estado de las carreteras

"El estado de las carreteras españolas no levanta cabeza. Entre 2005 y 2015 la conservación de la red viaria ha pasado del aprobado "por los pelos" al deficiente, calificación que se repite en estos diez años con una preocupante tendencia a la baja, aproximándose cada vez más a la línea roja del muy deficiente." Así comienza el informe sobre Necesidades de Inversión en Conservación realizado por la AEC en 2016. [28]

La Asociación Española de la Carretera empezó a realizar estudios sobre el estado del firme en 1985, con el fin de conocer el estado de pavimentación y equipamiento de las carreteras españolas, además de la inversión mínima necesaria para llegar a un nivel aceptable. Sólo en el periodo de 2013 a 2015, su mal estado ha incrementado en un 7%: un 2% en la red de carreteras que gestiona el estado, y un 9% en la que llevan los gobiernos autonómicos. Esto implica una inversión necesaria de 6.617 millones de euros. Aunque si este estado continúa empeorando, puede que en 2020 sea necesaria la reconstrucción de la mayor parte del firme.

Los valores del eje vertical de la Figura 2.7 corresponden a la calificación que le da AEC al estado del pavimento, siendo esta:

- 300 – 400 → Buena.
- 200 – 300 → Aceptable.
- 100 – 200 → Deficiente.
- 0 – 100 → Muy deficiente.

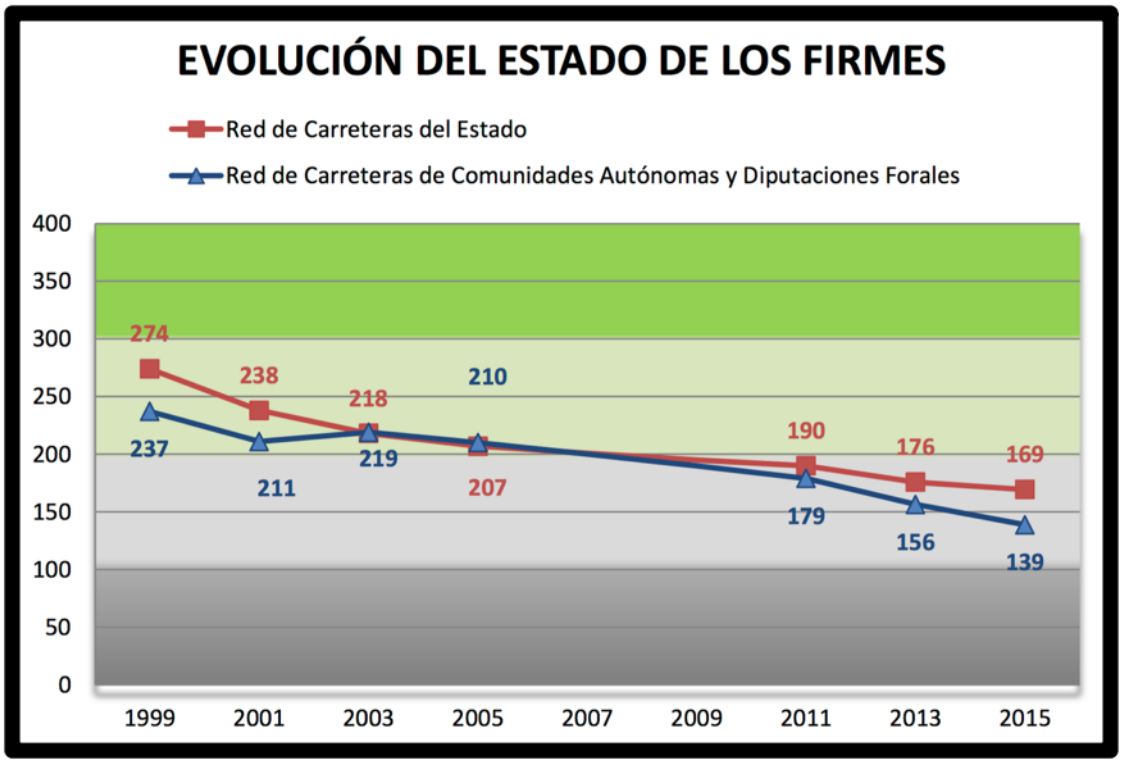


Figura 2.7. Evolución del estado de los firmes. [28]

Las consecuencias de este mal estado son directas sobre los conductores, tanto en la comodidad como en el cansancio de éste, además de un aumento de consumo y el deterioro del vehículo, pero sobre todo en lo referente a la seguridad vial.

Una carretera en mal estado aumenta el riesgo de sufrir un accidente por incomodidad del conductor, por deslizamiento del vehículo, o simplemente por evitar el desperfecto. En cuanto a las implicaciones que tienen sobre el vehículo, la Asociación Española de Fabricantes de Mezclas Asfálticas (Asefma), expone que las carreteras en mal estado aumentan el consumo de combustible en un 34% y disminuyen la vida útil del vehículo en un 25%. Por otro lado, este mal estado supone un incremento del 34% de las emisiones de gases de efecto invernadero. [29]

2.5. Entorno socio-económico

Actualmente existen algunas campañas de prevención que pretenden denunciar el mal estado del firme con la ayuda de los usuarios. A continuación, detallaremos ligeramente alguna de ellas:

- **Ponle Freno**: *Señales y carreteras en mal estado*, una campaña lanzada por el grupo Atresmedia, que busca la participación ciudadana para denunciar señales defectuosas o en mal estado y las carreteras en mal estado. Los ciudadanos tendrán que enviar una foto o vídeo junto con la localización del desperfecto y Ponle Freno posteriormente denunciará formalmente y reclamará su arreglo ante la administración correspondiente. [9]
- **Fundación CEA**: *Carreteras en mal estado o señales defectuosas*, es la campaña que lanza la fundación CEA con el fin de minimizar los riesgos o consecuencias de sufrir un accidente de tráfico. La fundación ha creado una plataforma *online* que permite a los ciudadanos denunciar los posibles casos. [10]
- **Automovilistas Europeos Asociados (AEA)**: *¡Díselo al ministro!, ¡Díselo al alcalde!, ¡Díselo al consejo autonómico!*, es el lema de AEA para establecer una mejor comunicación entre los automovilistas y la Administración, con respecto a la mejora de las infraestructuras. AEA considera que el estado de las carreteras tiene una implicación directa sobre la producción de accidentes. Es por esto por lo que ha creado su campaña de prevención. [11]

Por otro lado, dado el gran número de usuarios con teléfono móvil, existen también algunas aplicaciones como propuesta.

- **Street Bump:** Es una aplicación para iOS puesta a prueba en Boston, y creada para identificar baches a través del acelerómetro de los teléfonos de los usuarios. Los usuarios voluntariamente usarán la aplicación para recolectar información mientras conducen. La aplicación detectará un bache y mandará las coordenadas GPS a un banco de datos de la ciudad. [30]

Figura 2.8. Logo de la aplicación *Street Bump*. [30]

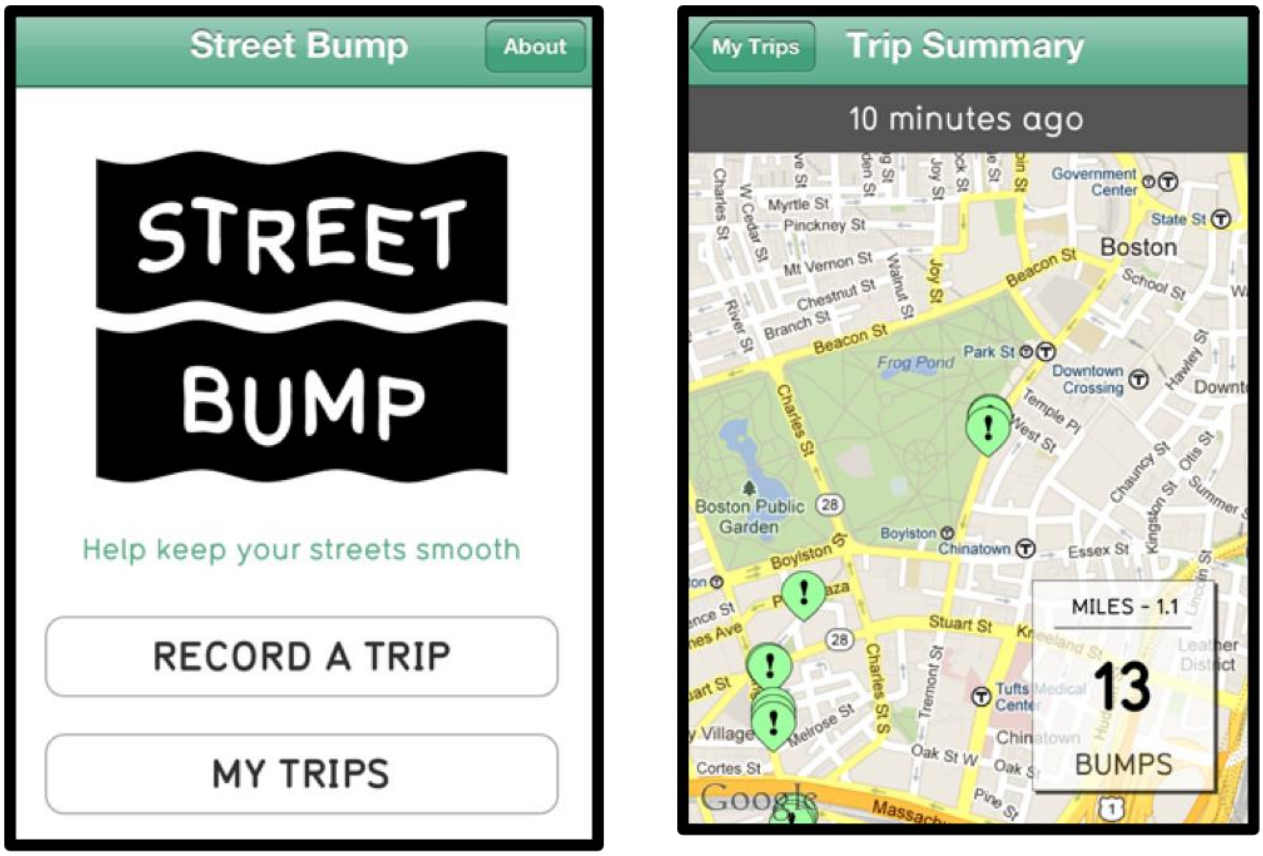


Figura 2.9. Aplicación *Street Bump*. [30]

- **Avisos Madrid:** Disponible tanto para iOS como para Android, es una aplicación que sirve como canal de comunicación de incidencias en la Comunidad de Madrid. En la aplicación encontramos diferentes modalidades para dar avisos tales como señalización o socavones en la calzada. A la hora de realizar un aviso a través de la aplicación podremos adjuntar una fotografía del desperfecto en cuestión, localizarlo en el mapa e incluir una pequeña descripción. [31]

Figura 2.10. Logo de la aplicación *Avisos Madrid*. [31]

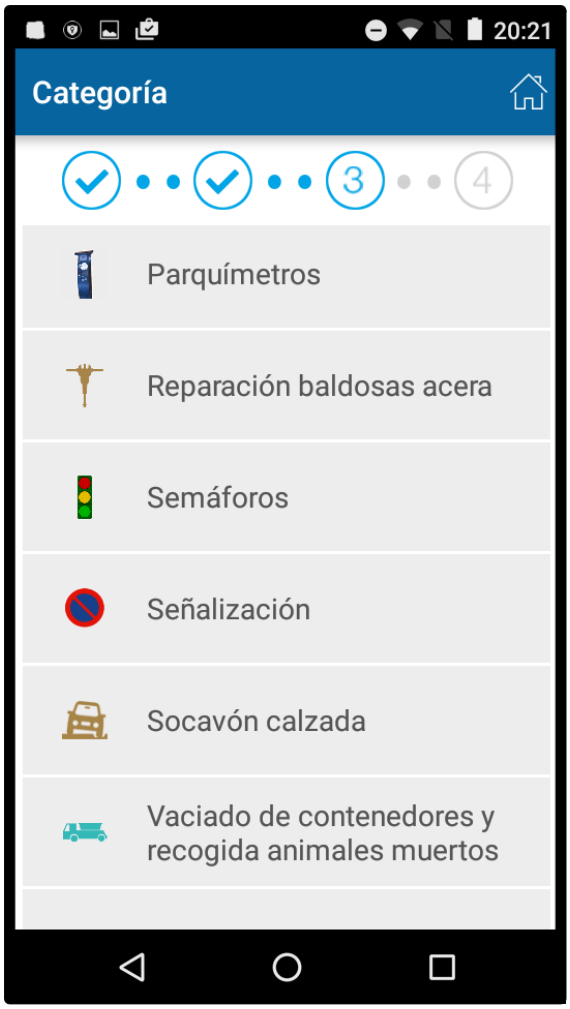


Figura 2.11. Aplicación *Avisos Madrid*. [31]

2.6. Marco regulador

Antes de comenzar a analizar la aplicación, estudiaremos las leyes vigentes y las normativas técnicas que atañen al proyecto.

La Constitución Española establece en su artículo 18 el derecho a la intimidad de las personas.

> 18.1. Se garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen.
>
> 18.4. La Ley limitará el uso de la Informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.

Más concreta es la Ley Orgánica 15/1999, del 13 de diciembre, de Protección de Datos de Carácter Personal, con entrada en vigor el 14/01/2000.

2.6.1. Ley Orgánica de Protección de datos de Carácter Personal

La Ley Orgánica de Protección de datos de Carácter Personal manifiesta en su artículo 1 que el objeto de esta ley es "garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar." [32]

Posteriormente, en su artículo 4 hace mención a la calidad de los Datos, expresando que sólo se podrán recoger aquellos datos personales que sean necesarios en relación con el ámbito y las finalidades determinadas para las que se hayan obtenido.

Una ley de protección de datos debe contar con el consentimiento del afectado, aspecto tratado en su artículo 6.1.

> "El tratamiento de los datos de carácter personal requerirá el consentimiento inequívoco del afectado".

Por supuesto, una vez obtenidos esos datos, debemos contemplar la seguridad de los mismos como lo refleja el siguiente artículo.

Artículo 9. Seguridad de los datos

> "El responsable del fichero, y, en su caso, el encargado del tratamiento, deberán adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que están expuestos, ya provengan de la acción humana o del medio físico o natural."

Por lo tanto, es necesario tomar medidas de seguridad para que la privacidad de los datos no se vea afectada.

2.6.2. Ley General de las Telecomunicaciones

La aplicación tendrá una proyección posterior en la que tendremos que tener en cuenta la Ley 9/2014, del 9 de mayo, de Telecomunicaciones. [33]

El objeto de esta ley es asegurar y avalar el secreto de las comunicaciones.

La aplicación, en un futuro, enviará información a través de las redes de comunicaciones, por lo que es importante tener en cuenta el artículo 43, que contempla el cifrado en las redes y servicios de comunicaciones electrónicas.

Artículo 43. Cifrado en las redes y servicios de comunicaciones electrónicas

> "1. Cualquier tipo de información que se transmita por redes de comunicaciones electrónicas podrá ser protegida mediante procedimientos de cifrado."
>
> "2. El cifrado es un instrumento de seguridad de la información. Entre sus condiciones de uso, cuando se utilice para proteger la confidencialidad de la información, se podrá imponer la obligación de facilitar a un órgano de la Administración General del Estado o a un organismo público, los algoritmos o cualquier procedimiento de cifrado utilizado, así como la obligación de facilitar sin coste alguno los aparatos de cifra a efectos de su control de acuerdo con la normativa vigente."

2.7. Conclusiones sobre el estado del arte

Para finalizar el estudio de los apartados anteriores, realizaré un breve análisis resumiendo los puntos clave para cada apartado.

Sobre los sistemas operativos, podemos concluir que existen dos principales, Android SO e iOS. La elección de Android se fundamenta en los datos expuestos, donde el 84% de los usuarios lo utilizan. Además, son de gran utilidad características como tratarse de una plataforma de software libre.

Sobre los sensores y módulos, podemos concluir que Android proporciona librerías para facilitar el uso de ellos. Que existe una gran variedad de los mismos. La elección del uso del acelerómetro, GPS y giroscopio fundamentado por los requisitos de la aplicación (dar una solución de bajo coste como vimos en los objetivos).

Sobre el estado de las carreteras, podemos concluir que el mal estado de las carreteras influye en un mayor deterioro del vehículo, lo cual como consecuencia aumenta las posibilidades de accidente.

Sobre el entorno socio-económico, podemos concluir que actualmente existen diversas soluciones a este mismo problema. Por lo que este proyecto deberá buscar una solución más fácil de utilizar (mediante la automatización) y más barata.

3. ANÁLISIS Y REQUISITOS DE USUARIO

Con las conclusiones y conocimientos adquiridos del estado del arte, estudiaremos los diferentes requisitos del proyecto. Para ello se realizará un análisis de los casos de uso, las historias de usuario y los requisitos de restricción del este TFG. Este apartado nos ayudará a identificar las necesidades del proyecto. En cada una de las secciones se explicará el porqué de la elección de esta metodología.

3.1. Casos de uso

Los casos de uso son una técnica utilizada para recoger situaciones que ocurren en el sistema, es decir, las posibles acciones que un usuario puede realizar sobre el sistema. Dicho de otra forma, se trata de identificar los requisitos funcionales del sistema, dividiéndolos en los distintos actores que interactúan con él.

Para representarlo utilizaremos dos actores, uno representará al usuario y otro al desarrollador de la aplicación, y su interacción con las distintas actividades. Se realizará un diagrama ilustrando los casos de uso. Además, de forma aclaratoria se utilizará una tabla, y una explicación, para cada caso de uso haciendo así más sencilla su comprensión.

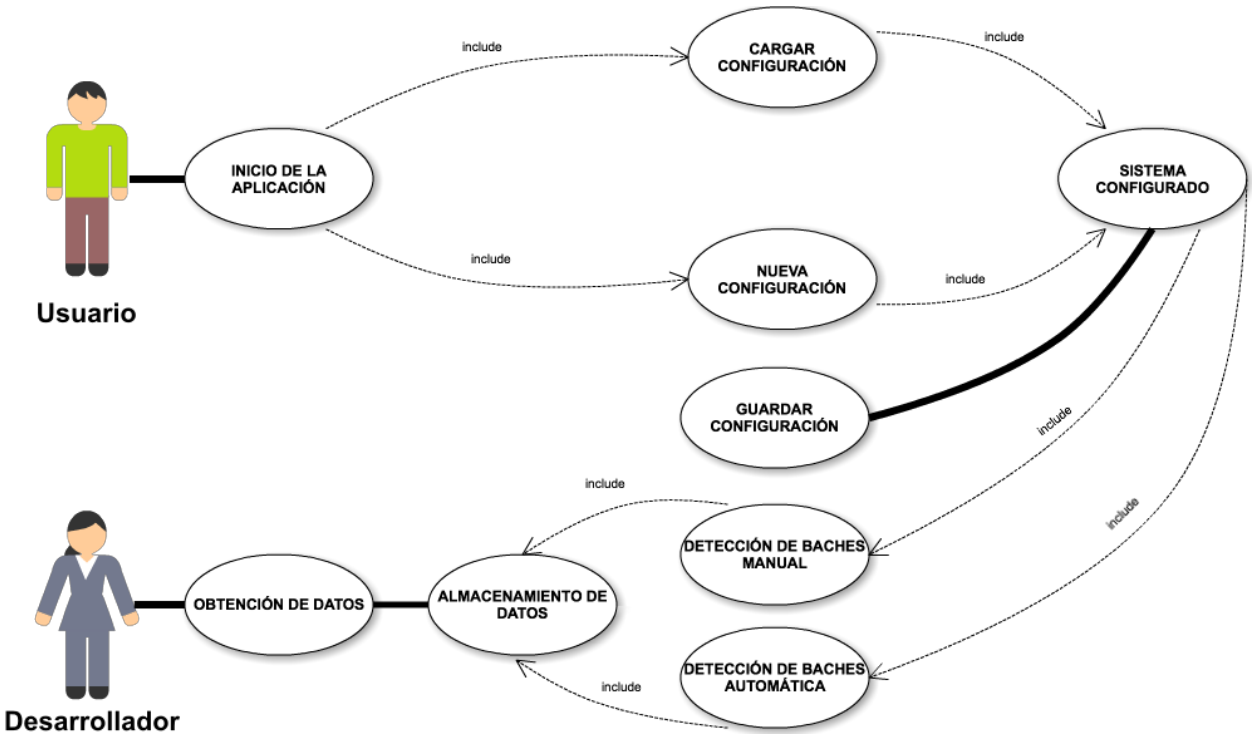


Figura 3.1. Diagrama de los casos de uso.

Las tablas de los casos de uso seguirán la siguiente plantilla:

Identificador	Código de identificación único para cada caso de uso. Seguirá el código CU – XX
Nombre	Nombre que se le da al caso de uso. No necesariamente tiene que ser unívoco pero si descriptivo.
Flujo normal	**Flujo alternativo**
Secuencia de acciones principales necesarias para llevar a cabo el caso de uso.	Secuencia alternativa al flujo normal de acciones que se pueden realizar.

Tabla 3.1. Plantilla de los casos de uso.

CU-01 Inicio: Este caso de uso se da cuando el usuario inicia la aplicación desde el menú de Android, y se muestra la pantalla de inicio. Es necesario que el usuario haya instalado la aplicación previamente. El actor es el usuario.

Identificador	CU – 01
Nombre	Inicio
Flujo normal	**Flujo alternativo**
• Abrir menú de Android. • Hacer clic en la aplicación.	En caso de que el usuario haya creado un acceso directo desde la pantalla principal: • Hacer clic en la aplicación.

Tabla 3.2. CU-01.

CU-02 Cargar configuración: Con la aplicación iniciada, el usuario podrá elegir la opción de cargar una configuración. Es necesario que haya una configuración previa almacenada. La aplicación queda configurada como resultado de esto. El actor es el usuario.

Identificador	CU – 02
Nombre	Cargar configuración
Flujo normal	**Flujo alternativo**
• Inicio de la aplicación (CU-01). • Cargar la configuración.	N/A

Tabla 3.3. CU-02.

CU-03 Nueva configuración: Con la aplicación iniciada, el usuario podrá elegir la opción de una nueva configuración. La aplicación queda configurada como resultado de esto. El actor es el usuario.

Identificador	CU – 03
Nombre	Nueva configuración
Flujo normal	**Flujo alternativo**
• Inicio de la aplicación (CU-01). • Nueva configuración. • Seguir los pasos indicados en pantalla.	N/A

Tabla 3.4. CU-03.

CU-04 Sistema configurado: Tras realizar una nueva configuración o una carga de esta, el sistema quedará configurado. A partir de este estado, la aplicación queda configurada para detectar baches. De forma indirecta el actor es el usuario que inició la configuración.

Identificador	CU – 04
Nombre	Sistema configurado
Flujo normal	**Flujo alternativo**
• Inicio de la aplicación (CU-01). • Cargar la configuración (CU-02). • Sistema configurado.	• Inicio de la aplicación (CU-01). • Nueva configuración (CU-03). • Sistema configurado.

Tabla 3.5. CU-04.

CU-05 Guardar configuración: Con la aplicación configurada, el usuario podrá elegir la opción guardar la configuración. La configuración queda almacenada para futuras cargas. El actor es el usuario.

Identificador	CU – 05
Nombre	Guardar configuración
Flujo normal	**Flujo alternativo**
• Inicio de la aplicación (CU-01). • Cargar configuración (CU-02). • Sistema configurado (CU-04). • Guardar configuración.	• Inicio de la aplicación (CU-01). • Nueva configuración (CU-03). • Sistema configurado (CU-04). • Guardar configuración.

Tabla 3.6. CU-05.

CU-06 Detectar baches manualmente: Con la aplicación configurada, el usuario podrá informar sobre baches de manera manual. Los datos podrán ser almacenados. El actor es el usuario.

Identificador	CU – 06
Nombre	Detectar baches manualmente
Flujo normal	**Flujo alternativo**
• Inicio de la aplicación (CU-01). • Cargar configuración (CU-02). • Sistema configurado (CU-04). • Detectar baches manualmente.	• Inicio de la aplicación (CU-01). • Nueva configuración (CU-03). • Sistema configurado (CU-04). • Detectar baches manualmente.

Tabla 3.7. CU-06.

CU-07 Detectar baches automáticamente: Con la aplicación configurada, el sistema podrá informar sobre baches de manera automática. Los datos podrán ser almacenados. De forma indirecta el actor es el usuario ya que inició la configuración, aunque es el sistema el que realiza la acción.

Identificador	CU – 07
Nombre	Detectar baches automáticamente
Flujo normal	**Flujo alternativo**
• Inicio de la aplicación (CU-01). • Cargar configuración (CU-02). • Sistema configurado (CU-04). • Detectar baches automáticamente.	• Inicio de la aplicación (CU-01). • Nueva configuración (CU-03). • Sistema configurado (CU-04). • Detectar baches automáticamente.

Tabla 3.8. CU-07.

CU-08 Almacenamiento de datos: Tras la información de datos sobre baches, estos serán almacenados en el sistema. De forma indirecta el actor es el usuario ya que inició la configuración, aunque es el sistema el que realiza la acción.

Identificador	CU – 08
Nombre	Almacenamiento de datos
Flujo normal	**Flujo alternativo**
• Sistema configurado (CU-04). • Detectar baches automáticamente (CU-07). • Almacenamiento de datos.	• Sistema configurado (CU-04). • Detectar baches manualmente (CU-06). • Almacenamiento de datos.

Tabla 3.9. CU-08.

CU-09 Obtención de datos: Tras el almacenamiento de datos sobre baches, estos serán analizados por el desarrollador. El actor es el desarrollador.

Identificador	CU – 09
Nombre	Obtención de datos
Flujo normal	**Flujo alternativo**
• Sistema configurado (CU-04). • Detectar baches automáticamente (CU-07). • Almacenamiento de datos (CU-08). • Obtención de datos.	• Sistema configurado (CU-04). • Detectar baches manualmente (CU-06). • Almacenamiento de datos (CU-08). • Obtención de datos.

Tabla 3.10. CU-09.

3.2. Historias de usuario

Las historias de usuario son una alternativa a los requisitos funcionales. Estas se caracterizan por estar escritas en una o dos frases, y por utilizar un lenguaje sencillo que sea entendible para el cliente. Habitualmente son utilizadas en metodologías de desarrollo ágil. En este TFG se ha decidido su uso por considerarse una manera sencilla y entendible, para todos los públicos, de explicar el funcionamiento de este proyecto. [34]

A partir de las historias de usuario redactadas, se irán desarrollando tareas más complejas, que explicarán la funcionalidad del proyecto.

Las siguientes tablas, Tabla 3.11 y Tabla 3.12, son un ejemplo de cómo estructuraremos las diferentes tareas e historias de usuario, para facilitar su lectura y desarrollo.

Identificador	Objetivos	Complejidad
Código de identificación único para cada historia de usuario. Seguirá el código HU – XX	Describe la funcionalidad que cumple el sistema	Dificultad que puede tener una historia de usuario. Sus valores son: Alta, media o baja.

Tabla 3.11. Plantilla de las historias de usuario.

Identificador	Tarea	Relacionado con
Código de identificación único para cada tarea. Seguirá el código T – XX.	Explica brevemente la tarea.	Código de la historia de usuario con la que está relacionada la tarea.

Tabla 3.12. Plantilla de las tareas.

Las historias de usuario definidas para este proyecto, están disponibles en la siguiente tabla:

Identificador	Objetivos	Complejidad
HU – 01	Se detectarán baches de forma automática.	Alta.
HU – 02	Se detectarán baches de forma manual.	Baja.
HU – 03	Se generará información de los baches localizados.	Media.
HU – 04	Se comprobará la configuración del sistema.	Media.

Tabla 3.13. Historias de usuario.

HU - 01 Se detectarán baches de forma automática:

La especificación de esta historia de usuario es:

- Los baches tienen que ser detectados de forma automática.
- Es necesaria una solución con un manejo sencillo.
- Se busca una solución de bajo coste, compatible con los dispositivos móviles.

Identificador	Tarea	Relacionado con
T - 01	Se investigan posibles soluciones para detectar baches de manera automática.	HU - 01
T – 02	Se proponen soluciones alternativas a las existentes actualmente, como, por ejemplo: utilizar el sensor del acelerómetro de los Smartphone.	HU - 01
T – 03	Se crea una aplicación para comprender el funcionamiento del acelerómetro.	HU - 01
T – 04	Se crea una aplicación que reoriente el acelerómetro.	HU - 01
T – 05	Se crea el código que permite la detección automática de baches.	HU - 01

Tabla 3.14. Tareas HU - 01.

HU - 02 Se detectarán baches de forma manual:

La especificación de esta historia de usuario es:

- Existirá una opción para detectar baches de manera manual.
- El usuario podrá informar manualmente de la existencia de baches. Se realizará a través de la interfaz de usuario.
- El usuario obtendrá información de los baches detectados manualmente. Se realizará a través de la interfaz de usuario.

Identificador	Tarea	Relacionado con
T - 01	Se crea un botón para que el usuario detecte baches de manera manual.	HU - 02
T – 02	Se informa por pantalla, cuando se detecta un bache, mostrando un mensaje.	HU - 02

Tabla 3.15. Tareas HU - 02.

HU - 03 Se generará información de los baches localizados:

La especificación de esta historia de usuario es:

- La aplicación ha de guardar la posición en la que se encuentran los baches localizados.
- La información recogida ha de ser almacenada en un fichero de texto.
- El fichero de texto generado ha de ser fácil de analizar.
- Se ha de mostrar por pantalla que un bache ha sido detectado.

Identificador	Tarea	Relacionado con
T - 01	Se crea una aplicación para comprender el funcionamiento del módulo GPS.	HU – 03
T – 02	Se informa por pantalla, cuando se detecta un bache, mostrando un mensaje.	HU – 03
T – 03	Se crea una aplicación para comprender el funcionamiento del almacenamiento de ficheros.	HU – 03
T – 04	Se define un formato adecuado y sencillo para el análisis de datos	HU – 03
T – 05	Se almacena la localización de los baches en ficheros de texto.	HU – 03

Tabla 3.16. Tareas HU - 03.

HU - 04 Se comprobará la configuración del sistema:

La especificación de esta historia de usuario es:

- La aplicación ha de informar al usuario del estado de la configuración.
- Se podrá guardar la configuración actual del sistema.
- Se podrá cargar una configuración almacenada previamente.
- El almacenamiento se realizará en ficheros de texto.

Identificador	Tarea	Relacionado con
T - 01	Se informa por pantalla, cuando la configuración haya finalizado de manera correcta, mostrando un mensaje.	HU – 04
T – 02	Se informa por pantalla, cuando la configuración haya finalizado de manera incorrecta, mostrando un mensaje.	HU – 04
T – 03	Se crea un menú adicional, para el guardado y cargado de configuraciones.	HU – 04
T – 04	Se crea una opción que permite guardar la configuración actual.	HU – 04
T – 05	Se crea una opción que permite cargar una configuración previamente guardada.	HU – 04
T – 06	Se mostrarán los valores del acelerómetro reorientado, para posibilitar la comprobación del correcto funcionamiento.	HU – 04

Tabla 3.17. Tareas HU - 04.

3.3. Requisitos de restricción

Tras haber analizado las historias de usuario, se realiza un listado de los requisitos de restricción. Los requisitos de restricción son de obligado cumplimiento, y en caso de no cumplirse alguno de ellos, la aplicación puede no funcionar de la forma deseada.

Para listar estos requisitos se han utilizado unas tablas que seguirán una versión reducida del estándar IEEE 830-1998 [35].

Identificador	Código de identificación único para cada requisito. Seguirá el código RR – XX.
Nombre	Nombre que se le da al requisito. No necesariamente tiene que ser unívoco pero si descriptivo.
Descripción	Descripción breve de la restricción que implica el requisito.

Tabla 3.18. Plantilla de los requisitos de restricción.

A continuación, se exponen la lista de requisitos con su correspondiente tabla.

Identificador	RR – 01
Nombre	Dispositivo Android
Descripción	El usuario deberá poseer un dispositivo móvil con sistema operativo Android, con una versión superior a *Jelly Bean 4.2*, para hacer uso de la aplicación.

Tabla 3.19. RR – 01.

Identificador	RR – 02
Nombre	Tamaño de los objetos
Descripción	Cualquier usuario, con carencias visuales leves, podrá leer e identificar claramente los mensajes y elementos que aparecen en la interfaz de usuario.

Tabla 3.20. RR – 02.

Identificador	RR – 03
Nombre	Características del dispositivo móvil
Descripción	Para el correcto funcionamiento de la aplicación, según lo analizado anteriormente, será necesario que éste posea el sensor acelerómetro y el módulo GPS.

Tabla 3.21. RR – 03.

Identificador	RR – 04
Nombre	Almacenamiento
Descripción	Será necesario disponer del espacio suficiente en el Smartphone para guardar los ficheros de datos.

Tabla 3.22. RR – 04.

Identificador	RR – 05
Nombre	Transmisión de datos
Descripción	El dispositivo deberá contar con una conexión USB o conexión a internet para poder transmitir los datos. Esto facilitará su posterior análisis.

Tabla 3.23. RR – 05.

Identificador	RR – 06
Nombre	Aceptar las condiciones de uso
Descripción	El usuario deberá aceptar las condiciones de uso del módulo GPS y del almacenamiento en la memoria del dispositivo.

Tabla 3.24. RR – 06.

4. DISEÑO Y DESARROLLO DE LA APLICACIÓN

A lo largo del capítulo se analizarán los aspectos generales del sistema, explicando los inicios de la aplicación, su desarrollo y la investigación realizada. Asimismo, se explicarán las bases teóricas que dan lugar a la implementación, se mostrarán diagramas de flujo que ayuden a entender la aplicación y se hablará de la logística interna.

4.1. Aspectos teóricos

Los aspectos teóricos descritos a continuación han sido usados de forma común con el trabajo de fin de grado "Aplicación Android para obtener información de tráfico en carreteras", Álvaro González Caballero, ya que son la base de ambos proyectos. [39] (aclaración al lector).

Antes de comenzar a explicar la arquitectura, y el desarrollo de la aplicación, es necesario introducir brevemente los conceptos teóricos básicos en los que esta se basa.

El interés de esta aplicación, radica en la fusión de dos elementos que pertenecen a la rutina diaria de millones de personas. Se descubre así la posibilidad de compatibilizar el uso del Smartphone, con el del vehículo, consiguiendo detectar baches en el firme. Esto ayudará a mejorar la seguridad de la carretera, lo que influye directamente en el número de accidentes (recordemos que los baches son una de las causas de accidente, como vimos en el capítulo 2). Todos estos factores nos invitan al desarrollo de esta aplicación.

Los conceptos teóricos, están basados en el estudio del artículo "*TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones*" [12]. A continuación, resumiré el contenido más destacado, que afecta a este proyecto.

TrafficSense es un artículo, que busca dar soluciones a diferentes problemas en la carretera, como pueden ser eventos de tráfico o baches, mediante el uso de los sensores de un Smartphone. En el caso de los baches, esto se realizará con el acelerómetro.

Como vimos en el apartado de los sensores, el acelerómetro es capaz de detectar aceleraciones en los distintos ejes cartesianos x, y, z. Recordamos que estos están definidos en un móvil de la siguiente manera (visto el terminal de frente y en formato vertical):

- Eje x: Apunta hacia la derecha, por lo que es horizontal.
- Eje y: Apunta hacia arriba, por lo que es vertical.
- Eje z: Apunta hacia el interior de la pantalla, por lo que es transversal.

A partir de este momento, será necesario también definir un sistema de referencia para el vehículo, estará definido por las letras X, Y, Z. Este sistema está ilustrado en la siguiente figura:

- Eje X: Apunta hacia el frente del coche.
- Eje Y: Apunta hacia la derecha del coche.
- Eje Z: Apunta hacia el suelo del coche.

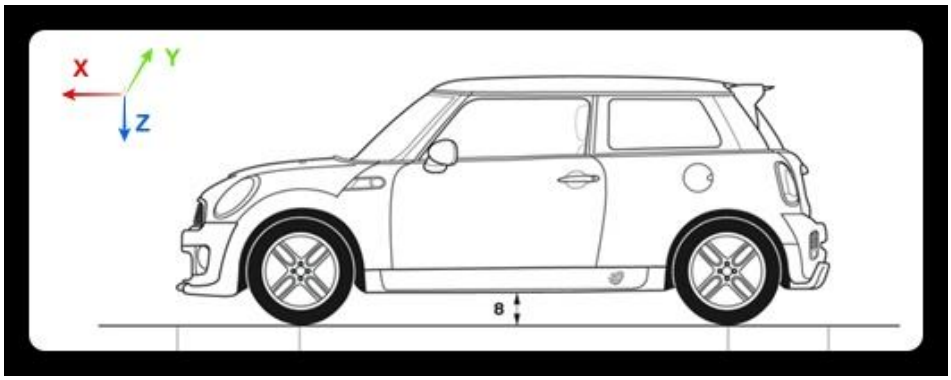


Figura 4.1. Ejes cartesianos del coche.

De esta forma podemos decir, que de producirse variaciones en el eje Z (correspondiente al coche), estas podrían deberse a baches o irregularidades en la vía. Por lo que ya tenemos nuestro sistema alternativo de detección mediante un sensor del Smartphone: el acelerómetro.

Una vez definida la solución, descubrimos que nos enfrentamos a un reto. Para que las medidas del eje Z del coche, tengan una correspondencia con las obtenidas en el dispositivo móvil, el vehículo y el dispositivo deberán estar correctamente alineados. Puesto que esto es una tarea casi imposible, debido a que no podemos obligar al usuario a llevar el móvil en una posición determinada, hemos de encontrar una solución.

Teniendo en cuenta que en un viaje de coche el dispositivo puede viajar en diversas posiciones (en el bolsillo del usuario, en un soporte para el coche, en un bolso o maleta, etc.), se deberá conseguir orientar de forma virtual el acelerómetro del móvil, para que al tomar las medidas se correspondan con el coche. Es decir, que mediante diversos cálculos conseguiremos que el eje X, Y, Z coincida con los ejes x', y', z', los ejes reorientados.

El artículo propone para ello como solución matemática, utilizar los ángulos de Euler. En concreto se utiliza la formulación Z – Y – Z. De esta forma será posible representar cualquier rotación, con los ángulos de *pre-rotación,* ϕ_{pre} sobre Z, *inclinación*, θ_{tilt} sobre Y, y *post-rotación*, ψ_{post}, sobre Z de nuevo.

La idea de la reorientación del acelerómetro, se basa en obtener muestras de la aceleración en determinados momentos, que permitan determinar cuáles serían los ejes virtuales del dispositivo (x', y', z'), correspondientes al vehículo (X, Y, Z). Para ello se buscan patrones en el

vehículo, que provoquen aceleración durante su uso en determinados ejes, y permitan fijar la reorientación. Por un lado, sabemos que, al mantenerse en reposo, por las características del acelerómetro, y la acción de la gravedad, se efectúa una fuerza constante de 9.8 m/s^2, en el eje Z. Por otro lado, la aceleración o frenado del vehículo en recto, provocará una aceleración en el eje X que podrá ser aprovechada. El eje Y podrá deducirse por la ortogonalidad del sistema de referencia.

Utilizando estas aceleraciones, se propone en el artículo dos diferentes momentos de toma de datos, y cálculo de los diferentes ángulos, pre-rotación, inclinación y post-rotación. Establecemos la notación $\boldsymbol{a_x, a_y, a_z}$ y $\boldsymbol{a_X, a_Y, a_Z}$ para referirnos a las aceleraciones en los ejes de dispositivo y vehículo, respectivamente.

4.1.1. Vehículo en reposo

Como indicamos anteriormente, mientras el vehículo permanece en reposo, tan sólo se medirá un valor de 9.8 m/s^2 en el eje Z (Es decir $\boldsymbol{a_Z = 9.8\ m/s^2}$). El ángulo de inclinación representará la diferencia de z con respecto de Z, y teniendo en cuenta la aceleración de la gravedad, obtenemos que:

$$\theta_{tilt} = \arccos\left(a_z\right)$$

Ecuación 4.1. Ecuación del ángulo de inclinación.

Una vez calculado $\theta_{\textbf{tilt}}$, calcularemos el ángulo de pre-rotación. Definiremos, que debido al efecto de la gravedad, $\boldsymbol{a_x}$ y $\boldsymbol{a_y}$ tendrán un valor distinto de cero. Obtenemos la siguiente ecuación:

$$\phi_{pre} = \arctan\left(\frac{a_y}{a_x}\right)$$

Ecuación 4.2. Ecuación del ángulo de pre-rotación.

A efectos de nuestra aplicación, debemos asegurarnos de que los datos $\boldsymbol{a_x, a_y, a_z}$, son tomados para el cálculo de los ángulos de pre-rotación e inclinación durante un periodo de reposo en el vehículo.

4.1.2. Vehículo en movimiento

Una vez definidos pre-rotación e inclinación, deberemos calcular post-rotación. Como ya vimos con anterioridad, podemos generar una aceleración en el eje X del vehículo, con tan solo una aceleración o un frenado. Puesto que los frenados, por naturaleza, se suelen producir de forma más aguda, serán los escogidos para la toma de datos. Por la definición del ángulo de post-rotación, sabemos que no se ve afectado por la fuerza gravitatoria. El artículo nos indica que usando la siguiente fórmula, podremos calcular con éxito el ángulo de post-rotación:

$$\psi_{post} = \arctan\left(\frac{-a_x \sin(\phi_{pre}) + a_y \cos(\phi_{pre})}{(a_x \cos(\phi_{pre}) + a_y \sin(\phi_{pre}))\cos(\theta_{tilt}) - a_z \sin(\theta_{tilt})}\right)$$

Ecuación 4.3. Ecuación del ángulo de post-rotación.

A efectos de nuestra aplicación, deberemos asegurarnos de tomar datos correctos, cuando se produzca un frenado. La metodología que utilizaremos será, mediante el GPS medir la velocidad, y en caso de que esta disminuya drásticamente, suponer que es un frenado y tomar datos del acelerómetro.

4.1.3. Cálculo de la matriz de rotación y transformación de valores

Con los valores obtenidos de los apartados 4.1.1 y 4.1.2, tendremos la suficiente información para calcular la matriz de rotación. El cálculo de esta matriz, nos permitirá transformar nuestro vector de aceleraciones $\boldsymbol{a_x, a_y, a_z}$ en el vector $\boldsymbol{a_X, a_Y, a_Z}$ tras la reorientación.

Para construir esta matriz, se han utilizado los conceptos teóricos del libro *"Dynamics of Tree-Type Robotic Systems"* [15], que, en su capítulo 3, trata sobre los ángulos de Euler, sus conjugaciones, y su manera de expresarlo en matrices. Encontramos que la matriz, llamada de rotación, ha de cumplir con ser ortogonal y tener determinante igual a 1. La matriz de rotación seguirá la siguiente ecuación:

$$R = R_\psi \cdot R_\theta \cdot R_\phi$$

Ecuación 4.4. Ecuación de la matriz de rotación.

Para obtener la matriz de rotación, necesitamos las matrices correspondientes a Rψ, Rθ y Rϕ.

$$R_\psi = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_\theta = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \quad R_\phi = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Ecuación 4.5. Ecuación de Rψ, Rθ y Rϕ.

El último paso a seguir para conseguir la reorientación del acelerómetro, es obtener los valores $\boldsymbol{a_x, a_y, a_z}$ a partir de los valores $\boldsymbol{a_X, a_Y, a_Z}$ medidos y la matriz de rotación calculada.

$$\begin{pmatrix} a_X \\ a_Y \\ a_Z \end{pmatrix} = R \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

Ecuación 4.6. Cálculo de los valores para el sensor reorientado.

4.2. Arquitectura del sistema

Siguiendo el patrón del modelo de arquitectura de software MVC (Modelo-Vista-Controlador), cuyo objetivo es facilitar las tareas de desarrollo de aplicación, así como sus posteriores tareas de mantenimiento, se ha decidido dividir la arquitectura de este sistema en tres partes: la capa vista, la capa modelo y controlador, y la capa del sistema operativo.

Capa vista: También conocida como interfaz de usuario, es la capa que presenta los datos del modelo. Los usuarios la utilizan para interactuar con la aplicación, por lo que esta capa contiene el código necesario para producir la salida del modelo que el usuario será capaz de interpretar.

En la capa vista, podemos encontrar objetos tales como:

- Un botón para detectar los baches de manera manual. Corresponde a la tarea T – 01 de la historia HU – 02.
- Un mensaje por pantalla indicando que se ha detectado un bache. Corresponde a la tarea T – 02 de la historia HU – 03.
- Un mensaje por pantalla indicando que la configuración se ha realizado de manera correcta o incorrecta. Corresponde a las tareas T – 01 y T – 02 de la historia HU – 04.
- Un menú adicional que permite guardar y cargar configuraciones. Corresponde a la tarea T – 03 de la historia HU – 04.
- Un texto por pantalla informando de los valores del acelerómetro reorientado. Corresponde a la tarea T – 06 de la historia HU – 04.

Capa modelo y controlador: Se ha decidido unificar la capa modelo y la capa controlador en base a las siguientes explicaciones.

La capa modelo, es la capa encargada de representar la lógica de la aplicación. Ésta, contiene el núcleo de la funcionalidad de la APP, por lo tanto, se encarga de recuperar los datos convirtiéndolos en elementos significativos para el sistema. En nuestro caso, el modelo corresponde a la obtención de datos a través del acelerómetro y el GPS, entre otros.

La capa controlador contiene el código necesario para responder a las acciones que se solicitan en la aplicación. Se utiliza de enlace entre vistas y modelos, pero sin manipular los datos de manera directa. Los

datos recogidos por el modelo, serán utilizados por el controlador, por ejemplo, para comunicar a la vista si se ha detectado un bache.
Los elementos de esta capa, como podemos observar en la Figura 4.2 son:

- Módulo de localización.
- Módulo de detección de movimientos.
- Módulo de reorientación.
- Módulo de almacenamiento de datos.
- Detección de baches.

El desarrollo de la aplicación está centrado en los elementos definidos en la arquitectura, por lo que se analizarán más exhaustivamente en las siguientes secciones.

Capa del sistema operativo: La capa del sistema operativo, en el caso que nos concierne, el sistema operativo Android, es imprescindible porque funcionará como intermediaria entre los elementos software y hardware. Además, nos proporciona las bibliotecas necesarias para facilitarnos la realización de tareas repetitivas de manera eficiente, como ya vimos anteriormente.

Esta capa se corresponde con la restricción RR – 01 explicada en el capítulo 3, donde se impone que es necesario que el terminal disponga de un sistema operativo Android.

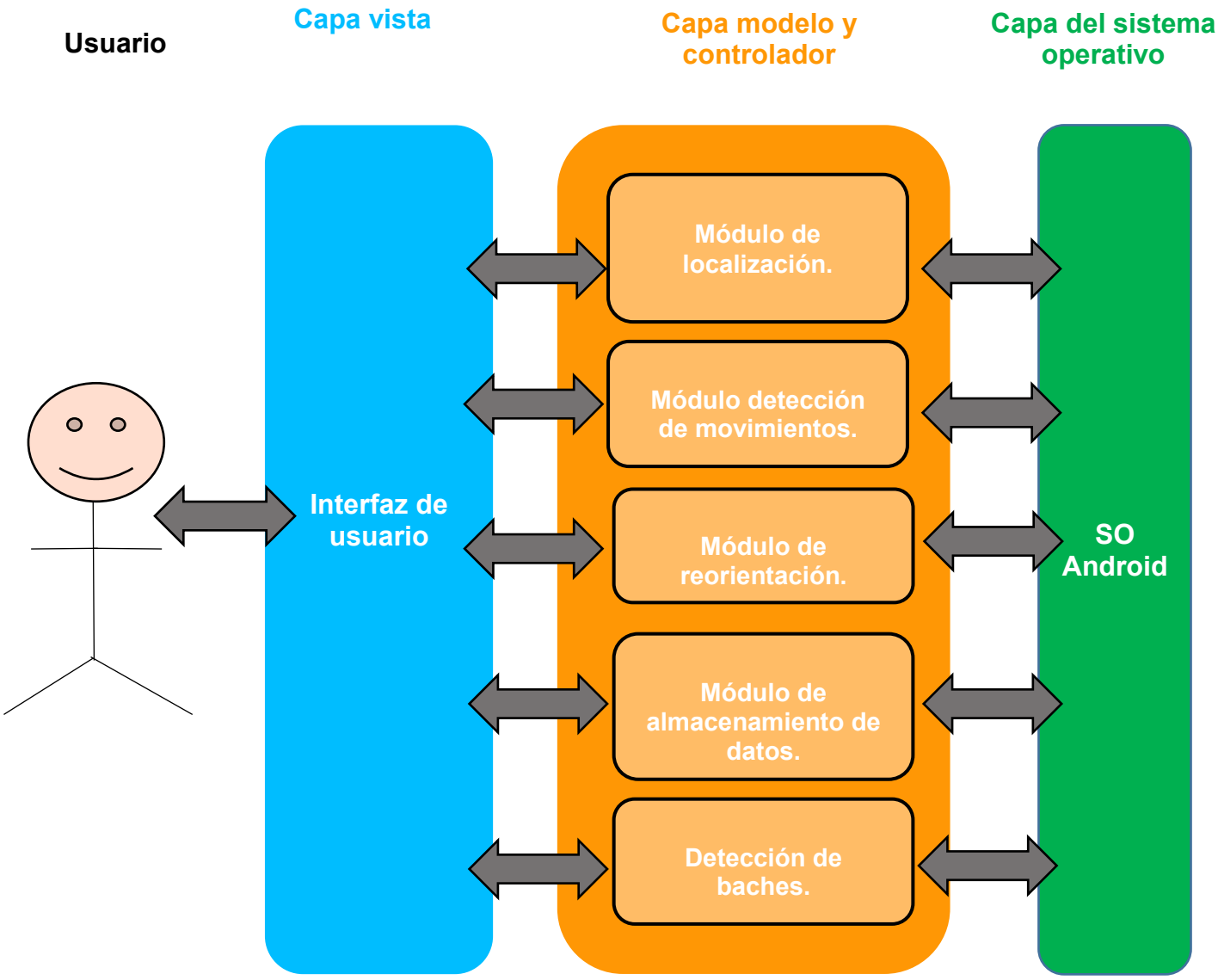


Figura 4.2. Arquitectura de diseño del sistema.

4.3. Módulos desarrollados

Los módulos expuestos a continuación han sido desarrollados en colaboración con el trabajo de fin de grado "Aplicación Android para obtener información de tráfico en carreteras", Álvaro González Caballero. [39] (aclaración al lector).

Partiendo de la arquitectura analizada en el apartado anterior, en esta sección describiremos los diferentes módulos desarrollados. Para ello, comenzaremos estableciendo una relación de cada módulo con su tarea e historia de usuario, para finalizar explicando el desarrollo del mismo.

4.3.1. Módulo de localización

La tarea T – 01 de la historia de usuario HU – 03, nos indica que es necesario crear una aplicación para establecer el funcionamiento del módulo de localización. Se utiliza para este módulo el GPS, por lo que adquiere una gran importancia en el desarrollo de esta aplicación, ya que se utilizará para establecer la ubicación de los baches detectados.

Debido al desconocimiento del uso de este sensor en un dispositivo Android, se decide crear una aplicación de prueba que permita entender el funcionamiento del mismo. El aspecto de la aplicación será el de la Figura 4.3.

Figura 4.3. Interfaz de la aplicación GPS.

En su interfaz gráfica se pueden observar diferentes campos: latitud, longitud, precisión, altitud y velocidad. Dado que nuestro objetivo final es detectar baches, necesitamos obtener su localización. Los parámetros más útiles serán los de latitud y longitud.

La clase principal para manejar los servicios de localización es *LocationManager*, disponible en el API de Android. Estos servicios permiten a las aplicaciones obtener actualizaciones periódicas de la

situación geográfica, o activar alguna funcionalidad cuando el dispositivo se encuentra próximo a una zona geográfica determinada. [14]

Las clases *Location* y *LocationListener*, también dentro del API de Android, nos permiten representar los datos de una localización geográfica o recibir eventos de cambios de localización, respectivamente.

4.3.2. Módulo de detección de movimientos

La tarea T – 03 de la historia de usuario HU – 01, nos indica que es necesario crear una aplicación para establecer el funcionamiento del módulo de detección de movimientos. Para este módulo, se decide utilizar el sensor de aceleración. El acelerómetro será una pieza fundamental en la detección de baches (como ya vimos en la sección 4.1).

Como ocurría con el GPS, se desconoce el funcionamiento de este sensor dentro del Sistema Operativo Android. Para familiarizarse con el mismo, se decide realizar una aplicación básica de prueba. El aspecto de esta aplicación está en la Figura 4.4.

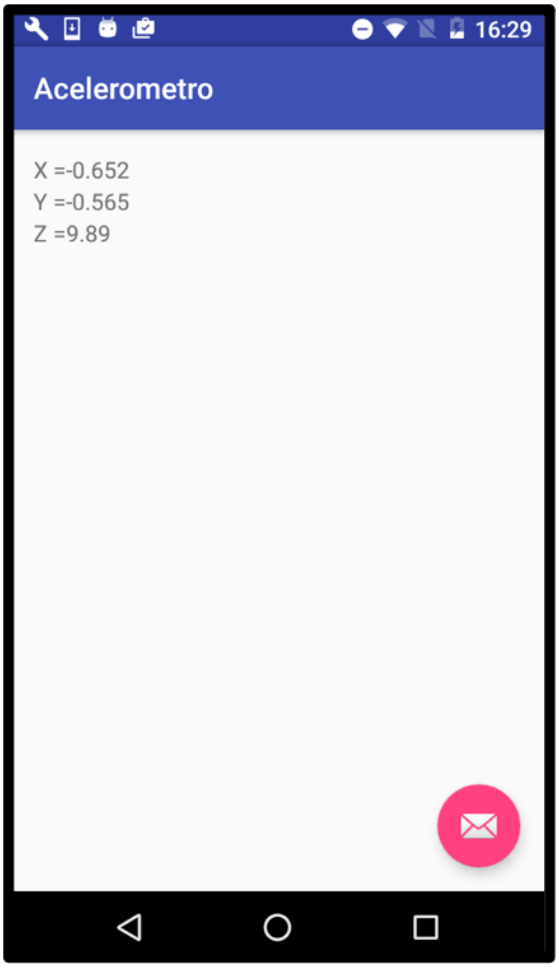


Figura 4.4. Interfaz de la aplicación acelerómetro.

En la interfaz gráfica, podemos observar los valores de aceleración para los ejes x, y, z, que se corresponden con el sistema cartesiano, explicado en la sección 4.1, para el dispositivo móvil. Los valores mostrados están expresados en la unidad del sistema métrico internacional, m/s^2.

Podemos manejar las opciones del acelerómetro a través de diferentes clases Android, pero la principal es *SensorManager*. A través de ella es posible acceder a los diferentes sensores instalados en el dispositivo. Además, las clases *Sensor* (instancia que representa a los diferentes sensores), *SensorEvent* (clase que representa un evento de un sensor, y que cuenta con la información del mismo) o *SensorEventListener* (permite recibir notificaciones de un sensor, cuando hay nuevos datos disponibles), permiten configurar de forma amplia el comportamiento de nuestro acelerómetro (y de otros sensores). [14]

4.3.3. Módulo de reorientación

Tras el análisis del artículo *TrafficSense* efectuado en la sección 4.1, sabemos que es necesario reorientar el acelerómetro, hasta que adopte los ejes del vehículo. Esta acción está requerida en la tarea T – 04 de la historia de usuario HU – 01.

Para esta tarea, partiremos de lo especificado en el artículo, y el funcionamiento básico obtenido en el módulo de detección de movimientos. En base a ello, creamos el siguiente diagrama de flujo del funcionamiento en general.

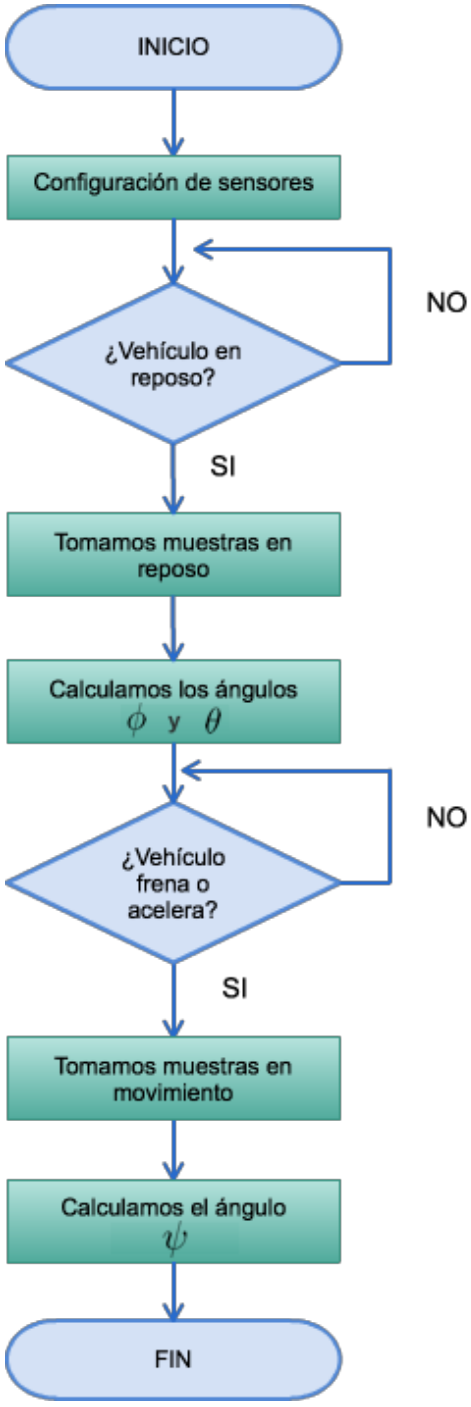


Figura 4.5. Diagrama de flujo del módulo de reorientación.

Una vez realizado el diagrama, y comprendidas las necesidades, se procede a desarrollar la aplicación. Para cumplir con los pasos descritos en el diagrama de flujo, se decide añadir un botón “Calibrar” en la interfaz de usuario. Este botón, al ser pulsado, iniciará la secuencia de calibrado. Durante un periodo de tiempo (entre 5 y 10 segundos) tomará datos con el vehículo en reposo, y calculará los ángulos correspondientes. Con una segunda interacción, el botón iniciará la fase de toma de datos en movimiento. De forma automática, gracias al uso del GPS, detectará una

posible frenada del vehículo, y tomará medidas del acelerómetro durante los siguientes segundos (entre 2 y 5).

Para finalizar, la aplicación con los datos obtenidos, calculará la matriz de rotación. En este punto, se podrá comenzar a reorientar los ejes del Smartphone, transformándolos a los ejes del coche. La siguiente figura representa la interfaz de usuario implementada.

Figura 4.6. Interfaz gráfica de la aplicación de reorientación.

4.3.4. Módulo de almacenamientos de datos

En la tarea T – 03 de la historia de usuario HU – 03, se nos indica que es preciso crear una aplicación, para comprender el funcionamiento del módulo de almacenamiento de datos. Para el almacenaje de datos, se utilizan ficheros de texto. Esta parte será de gran utilidad, puesto que nos permitirá almacenar los datos sobre la localización de los baches de forma local en el dispositivo. Además, podrá guardar una reorientación realizada o cargar una previamente guardada, a través de un menú de opciones tal y como se indica en las tareas T – 04 y T – 05 de la HU – 04.

Para poder recopilar información sobre los baches, se decide crear una pequeña aplicación, que almacene los datos introducidos por el usuario a través de la interfaz. Esta sirve como primera toma de contacto

con el almacenamiento de datos y el uso de los ficheros. La interfaz gráfica de la aplicación se muestra en la siguiente figura.

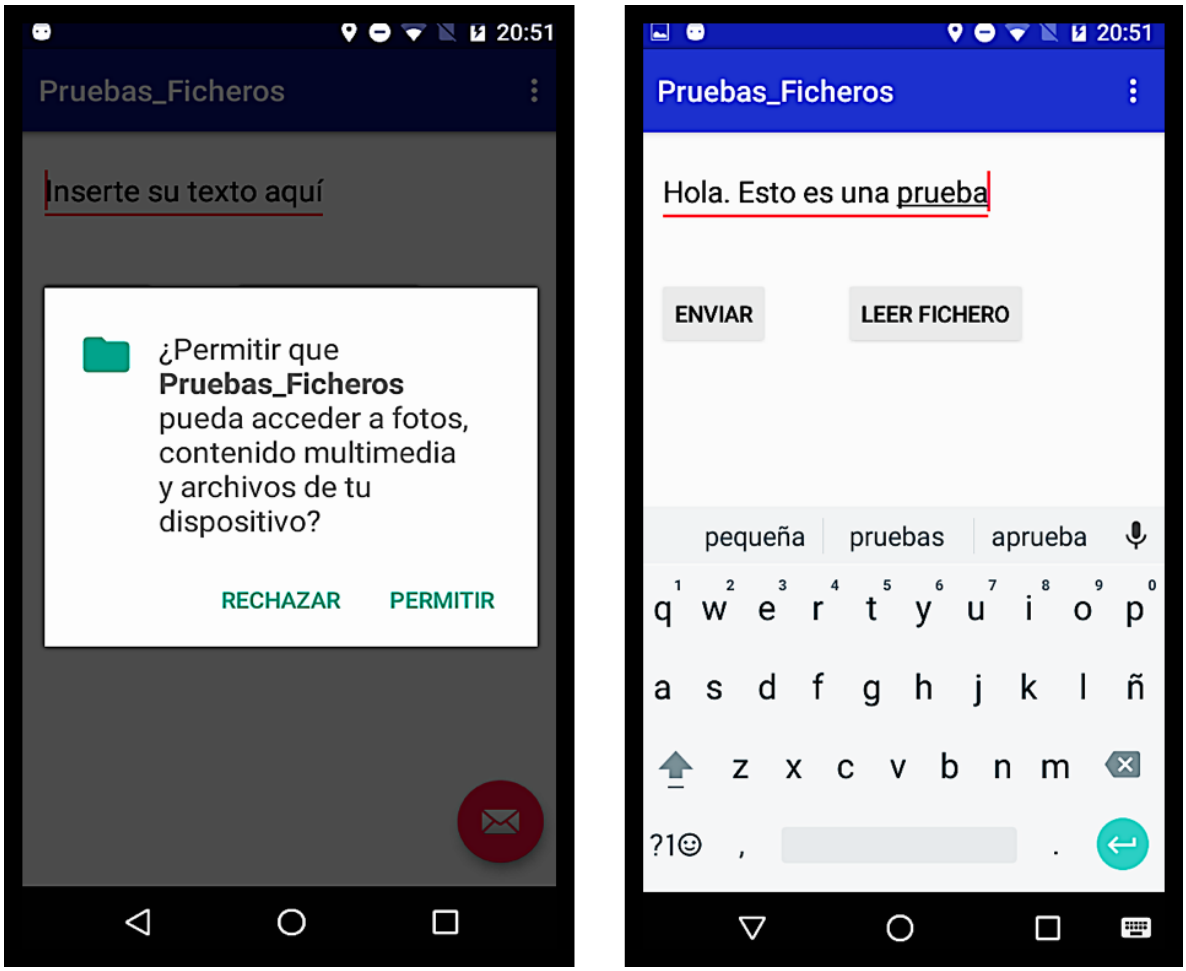


Figura 4.7. Interfaz gráfica de la aplicación de almacenamiento de datos.

La aplicación creada, almacena en la memoria interna del teléfono un fichero con los datos introducidos por pantalla al pulsar el botón "Enviar". Para ello, es necesario primero solicitar los permisos de almacenamiento al usuario (requisito de restricción RR – 06). Las clases que nos proporciona el API de Android para el manejo de ficheros son *OutputStreamWriter, FileOutputStream* y la clase *File*. [14]

4.3.5. Clases de los módulos

A raíz de los módulos detallados en los apartados anteriores, se generan una serie de métodos, que nos permiten realizar tareas que de otra manera serían repetitivas, creando una aplicación más eficiente.

Estos métodos están divididos en diferentes clases Java. Aunque la aplicación cuenta con un total de tres clases, las clases *My Application* y *Operaciones* se han realizado de forma conjunta para el TFG *"Aplicación Android para obtener información de tráfico en carreteras"* [39] tal y como se indicó al inicio de este apartado. (aclaración al lector)

Las clases que conciernen al desarrollo de los módulos son las siguientes:

My Application: La clase *My Application* almacena las variables globales de la aplicación. Además, contiene los métodos *get* y *set* que permitirán la lectura y escritura de estas variables en otras clases de la APP. Hereda de la clase *Application* del API de Android, y su método onCreate(), se ejecuta en primer lugar una única vez al abrirse la APP.

Operaciones: Para el análisis de la clase *Operaciones*, se realizará una división en módulos. Indicando así los métodos utilizados para cada uno.

- Reorientación: Este módulo consta de los siguientes métodos.

Método	Descripción
calibradoReposo()	Método que calcula la calibración del acelerómetro en reposo y obtiene las medidas necesarias para calcular ϕ_{pre} y θ_{tilt}.
calibradoMovimiento()	Método que se encarga de la calibración del acelerómetro en movimiento. Obtiene las medidas necesarias para el posterior cálculo de ψ_{post}.
calcularMatrizRotación()	Método que se encarga de calcular la matriz de rotación con los ángulos ϕ_{pre}, θ_{tilt} y ψ_{post}.
calcularVectorMediana()	Método que calcula la mediana de un vector.
calcularMediana()	Método que calcula la mediana de un array de valores.
ordenarArray()	Método que ordena de mayor a menor los valores del array.
productoMatrizVectorColumna()	Método que calcula el producto de una matriz por un vector columna. El resultado es un vector columna
productoVectorFilaMatriz()	Método que calcula el producto de un vector fila por una matriz. El resultado es un vector fila
producto()	Método que realiza el producto entre dos matrices
normalizarVector()	Método que normaliza un vector.
calcularMedia()	Método que dada una lista de valores calcula su media.

Tabla 4.1. Métodos del módulo de reorientación.

- Almacenamiento: Este módulo consta de los siguientes métodos.

Método	Descripción
escribirFichero()	Método que prepara el fichero para empezar a recoger los datos. Comprueba y crea el fichero.
guardarCalibrado()	Método que guarda los valores de ϕ_{pre}, θ_{tilt} y ψ_{post} para guardar el calibrado.
cargarConfiguracion()	Método que carga los valores de ϕ_{pre}, θ_{tilt} y ψ_{post} para cargar el calibrado.

Tabla 4.2. Métodos del módulo de almacenamiento.

La clase *Main*, que es de autoría propia, se explicará en capítulos posteriores.

4.4. Desarrollo de la aplicación final

En este apartado, se realizará una explicación detallada del desarrollo de la aplicación final. Se partirá de los módulos explicados anteriormente, se crearán distintos prototipos para la toma de datos, y finalmente se creará la aplicación que detecta los baches de forma automática. Para documentar este desarrollo, añadiremos una explicación sobre la interfaz de usuario, y un diagrama de flujo de la aplicación completa.

4.4.1. Obtención y análisis de datos

Antes de comenzar con el diseño completo de la aplicación final, se implementa un primer prototipo que únicamente registrará los eventos manuales. Además, aprovechando el módulo de almacenamiento de datos, se guardarán datos relacionados con el valor del acelerómetro, para el eje Z de forma continua, y para los eventos manuales, cuando estos ocurran.

Con este prototipo, cumpliremos la HU – 02 sobre la detección de baches de forma manual, y además estaremos consiguiendo que la aplicación tome datos que podremos analizar posteriormente, cumpliendo así con la tarea T-04 de la HU – 03 (sobre la definición de un formato para el almacenamiento de datos).

De forma gráfica, la interfaz mostrará nuevos controles. Se mantiene parte de la estructura creada en el módulo de calibración, con elementos como: los datos de aceleración para X, Y, Z, el botón de calibrar, y una serie de mensajes.

Los nuevos controles añadidos, son dos botones, que harán posible la detección de eventos manuales. Reciben el nombre de “Bache” y “Resalto”. La diferencia entre ambos conceptos será explicada junto con el análisis de los datos obtenidos.

Figura 4.8. Interfaz de la aplicación de recopilación de datos.

Inicialmente, se tomaron muestras de los datos de manera manual, intentando siempre seguir la misma ruta para así poder definir un umbral más preciso. Los datos se almacenaron en un único fichero, siguiendo el formato de la Figura 4.9, donde el primer campo es el tiempo, el segundo campo es el valor del eje Z medido por el acelerómetro y el tercero es la velocidad.

```
1   1466618277404, 8.819233, 34.174877;
2
3   1466618277420, 8.380763, 35.45345;
4
5   1466618277440, 12.343988, 35.45345;
6
7   ---------------------- BACHE!!!
8
9   1466618277459, 11.731902, 32.89446;;
10
11  1466618277479, 10.453917, 32.89446;
12
13  1466618277500, 9.4263525, 32.89446;
```

Figura 4.9. Formato inicial del almacenamiento de datos.

Para el análisis de los datos se decide utilizar el software MATLAB [36], puesto que con él se pueden representar los datos en un gráfico y ver la variación del eje Z.

Los datos almacenados de esta forma no eran útiles para su posterior análisis, ya que cuando MATLAB se encontraba con la línea en la que se indicaba un bache, se producía un error.

Para solucionar este problema, se decidió crear dos ficheros separados para almacenar la información. Por un lado, el primero recibe el nombre de *datos*, y en él se almacenan los datos del acelerómetro capturados de forma continua, incluyendo el tiempo, el valor de a_Z y la velocidad. En el segundo fichero, con el nombre *baches*, únicamente se almacenarán los datos sobre baches y badenes creados manualmente, incluyendo el tiempo, y un 2 si es un bache, o un 9 si es badén. En la Figura 4.10 se ve un ejemplo de estos ficheros.

```
datos = [
1468325545628, 9.267564, 60.678337;

1468325545631, 8.403482, 60.678337;

1468325545651, 11.469065, 60.678337;]
```

```
baches = [
1468325518766, 2;

1468325545309, 2;

1468325545678, 2;

1468325549107, 0;]
```

Figura 4.10. Formato final del almacenamiento de datos.

Puesto que el tiempo sólo se vería afectado unas milésimas de segundo (el tiempo que tarda el cuerpo humano en reaccionar), se podrían representar en MATLAB los datos de estos ficheros creando un código y añadiendo unos pequeños retoques, como por ejemplo la eliminación de las últimas muestras creadas por Android (las muestras con valor 0), para permitir cerrar el fichero de datos de manera correcta.

A continuación, podemos observar los datos obtenidos de la toma de datos manual, y la representación de los mismos.

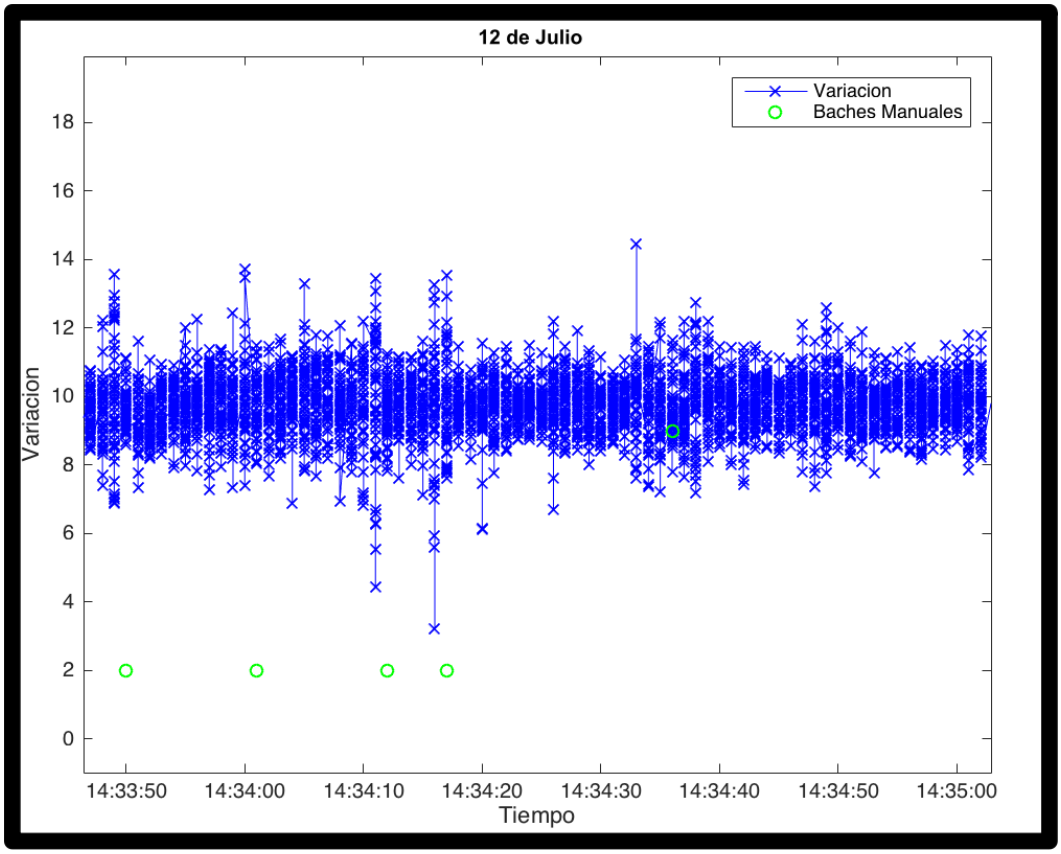


Figura 4.11. Gráfico de los datos de baches y badenes obtenidos manualmente.

Como se puede apreciar en la Figura 4.11, existen dos niveles en los que se detecta una variación en el eje Z. Uno se encuentra en 2 y otro en 9. Esto nos permitirá acotar el nivel de variación (umbral) cuando encontramos un desperfecto en el pavimento, sin tener falsos positivos como podría ser un badén en ciudad. Para ello, es preciso definir los siguientes conceptos:

- **Bache:** Según la *Real Academia Española* (RAE): "Hoyo o desigualdad en el pavimento de las calles, carreteras o caminos." [37]
- **Resalto:** Según la RAE: "También denominado badén o reductor de velocidad. Es un obstáculo artificial alomado que se pone de través en la calzada para limitar la velocidad de los vehículos." [37]

Figura 4.12. Definición gráfica de bache.

Figura 4.13. Definición gráfica de badén.

4.4.2. Aplicación final

En este apartado, se expondrá el desarrollo de la aplicación final, analizando las características añadidas respecto a la aplicación creada en el apartado anterior.

Los ajustes realizados, detección automática de baches, cumplen con la HU – 01, en concreto con la tarea T – 05, que especifica, el requerimiento de desarrollar un código para la detección automática de desperfectos en el pavimento.

La funcionalidad añadida que completa el desarrollo de la aplicación final, consiste en la creación de un algoritmo capaz de detectar variaciones en a_Z, diferenciando, de entre todos los datos, aquellos eventos que corresponden a baches. De forma adicional, se almacenarán estos eventos en un nuevo fichero con el nombre *baches_auto*, únicamente almacenando los datos sobre baches detectados automáticamente. Para ello se incluirá la marca de tiempo, el indicador y la localización obtenida del GPS. La localización será añadida también en el fichero ya existente con el nombre *baches* (que almacena los baches detectados manualmente).

Del análisis de los datos del apartado 4.4.1, se establece un umbral y se procede a probar la aplicación final. A continuación, una muestra de los datos obtenidos.

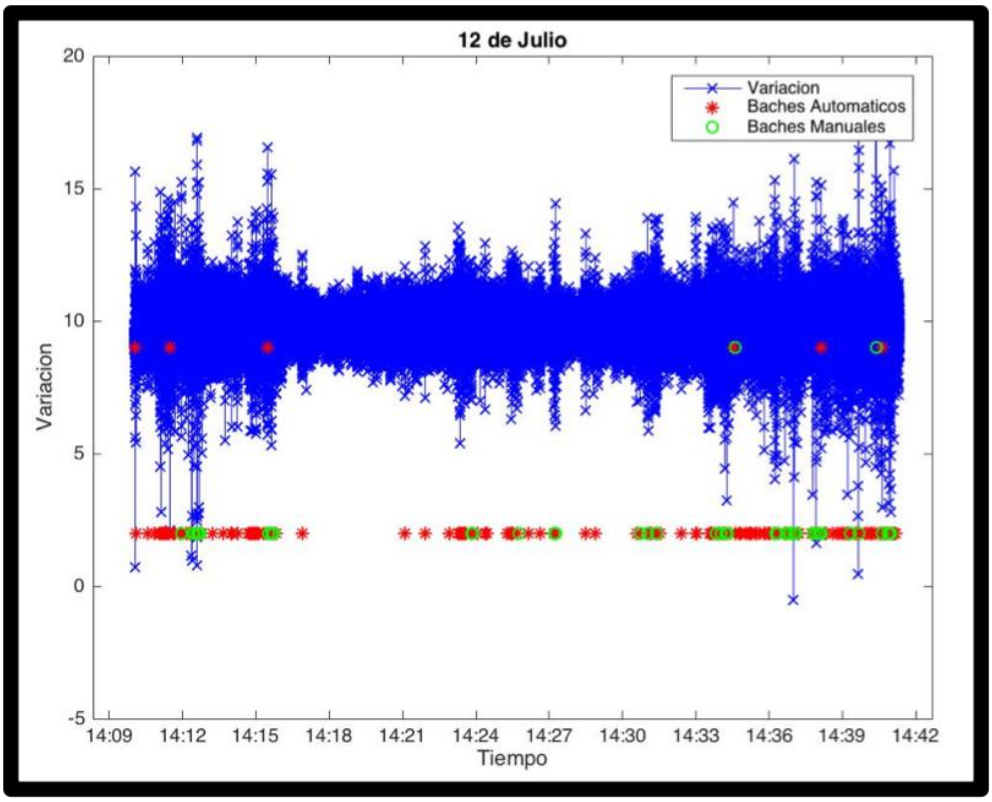


Figura 4.14. Gráfico de los datos obtenidos manualmente y automáticamente, 12 de Julio.

Como se puede observar, en la Figura 4.14, el valor inicial se acotó a una variación del eje Z de ±13 m/s^2. Si comparamos los datos obtenidos manualmente, círculos en verde, con los datos obtenidos de los baches automáticos, asteriscos en rojo, observamos que la detección automática no es precisa. Será necesario un reajuste. Estudiando la gráfica se

considera que un posible valor de acotación podría ser ±14,5 m/s^2 de variación. Se prueba nuevamente la aplicación y en la Figura 4.15 se puede observar el resultado.

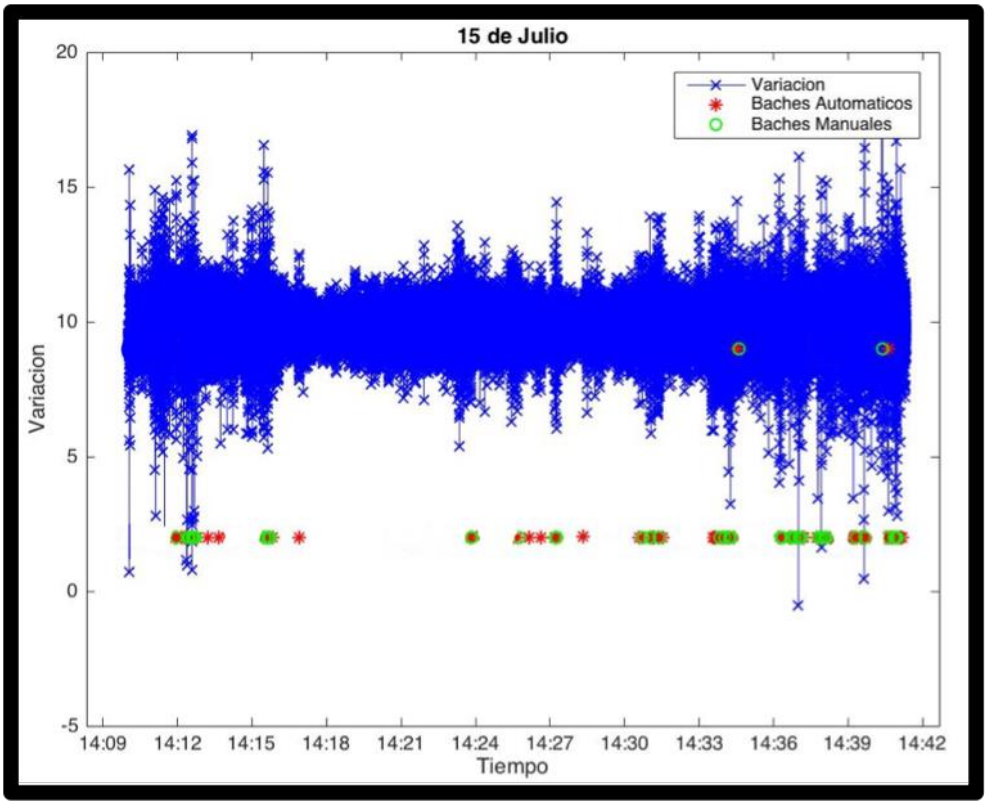


Figura 4.15. Gráfico de los datos obtenidos manualmente y automáticamente, 15 de Julio. Segunda acotación.

Analizando nuevamente la gráfica obtenida, podemos concluir que, a pesar de obtener algún falso positivo, el funcionamiento en general es mucho más preciso que la gráfica examinada anteriormente.

El nuevo código que ha sido generado para implementar la aplicación final, se implementa en la clase *Main*. Esta clase es de autoría propia.

4.4.3. Interfaz gráfica

El principal uso de la interfaz de usuario consiste en proporcionar un entorno visual, para permitir la comunicación con el sistema operativo. Por ello, es necesario que dicha interfaz sea sencilla, que no ofrezca dificultad, e intuitiva, que permita su comprensión instantáneamente, sin necesidad de razonamiento.

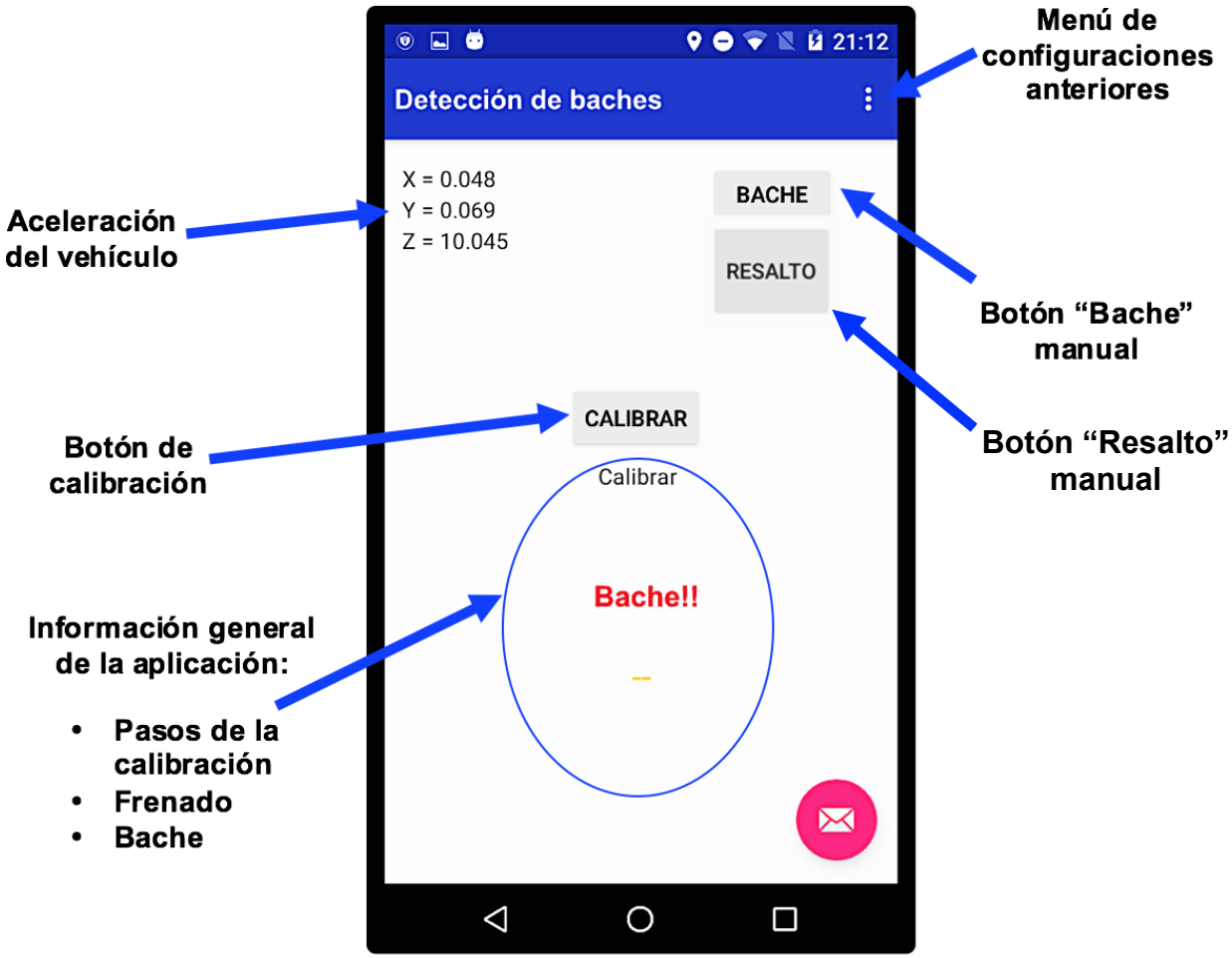


Figura 4.16. Interfaz de usuario principal.

Aceleración del vehículo: Indica el valor de la aceleración de los ejes a_x, a_y, a_z del vehículo.

Botón "Calibrar": Permite iniciar una nueva calibración siguiendo los pasos indicados.

Información general de la aplicación: Informa sobre los pasos a seguir para la calibración y si se ha producido una frenada o se ha detectado un bache.

Menú de configuraciones anteriores: Permite ver un menú desplegable para seleccionar la opción de cargar un calibrado o guardar un calibrado.

Botón "Bache" manual: Permite al usuario almacenar un evento de bache manualmente.

Botón "Resalto" manual: Permite al usuario almacenar un evento de badén manualmente.

Cargar calibrado: Permite cargar el último calibrado que ha sido guardado previamente.

Guardar calibrado: Guarda el calibrado actual para su posterior carga.

Figura 4.17. Interfaz de usuario con menú desplegable.

4.4.4. Diagramas de flujo

El diagrama de flujo, también conocido como diagrama de actividades, se utiliza en disciplinas como programación, ya que representa los diferentes estados que surgen cuando el usuario realiza una acción, y nos ayuda a comprender mejor la funcionalidad del proyecto.

Para su realización, se ha seguido el modelo UML (Lenguaje Unificado de Modelado) que consta de los siguientes elementos:

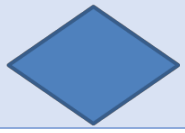
Elemento	Descripción	Representación
Nodo de Inicio/Fin	Indica el inicio o el fin de la actividad. Se representa con un circulo o una elipse.	
Nodo de actividad	Indica la ejecución de una o más actividades o procedimientos. Se representa con un rectángulo	
Nodo de decisión	Formula una pregunta o cuestión. Se representa con un rombo.	
Flujo de control	Indica la siguiente actividad en el flujo de control. Se representa con una flecha.	

Tabla 4.3. Elementos del diagrama UML.

Antes de comenzar a realizar los diferentes diagramas de flujo de la aplicación, es necesario comprender que Android es un sistema basado en *Listener* (o escuchadores). Estos *Listener* crean un flujo paralelo al principal cuando se produce un evento de algún tipo. Por esta razón, para facilitar el entendimiento de la secuencia de actividades, se van a realizar los siguientes diagramas de flujo:

- **Flujo principal:** Es el encargado de iniciar y finalizar la aplicación. Si este flujo no está iniciado, el resto de flujos no podrán lanzarse. Suele contener la inicialización de las variables más importantes, y la configuración de los sensores o módulos de Android necesarios para el funcionamiento de la aplicación.
- **Flujo botón "Bache":** Su función es detectar si se ha pulsado el botón "Bache", indicando un evento manual. De ser así se registrará un evento en el fichero correspondiente.
- **Flujo botón "Resalto":** Su función es detectar si se ha pulsado el botón "Resalto", indicando un evento manual. De ser así se registrará un evento en el fichero correspondiente.
- **Flujo botón "Calibrar":** Su función es detectar si se ha pulsado el botón "Calibrar", indicando el deseo del usuario de realizar una calibración. Si esto ocurre, se realizarán los pasos descritos en el diagrama de flujo correspondiente a la sección 4.3.3.
- **Flujo de detección automática de baches:** Es el flujo que proporciona el funcionamiento principal de la aplicación. Comprueba si el acelerómetro está calibrado, para luego utilizando el mismo, tomar valores de aceleración, y comprobar si existe algún bache. De comprobarse la existencia se generará una entrada en el fichero correspondiente, y una notificación al usuario.

Las siguientes figuras corresponden a los diferentes flujos que acabamos de describir.

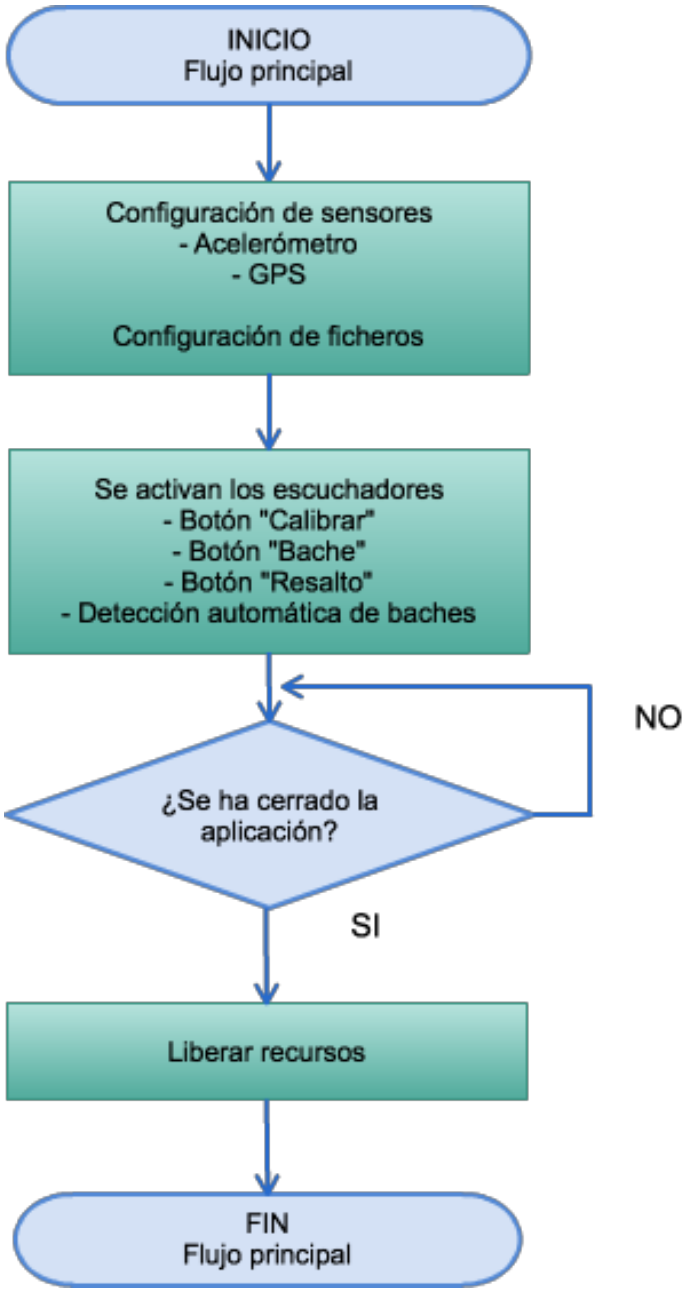


Figura 4.18. Diagrama del flujo principal.

Figura 4.19. Diagrama de flujo del botón "Bache".

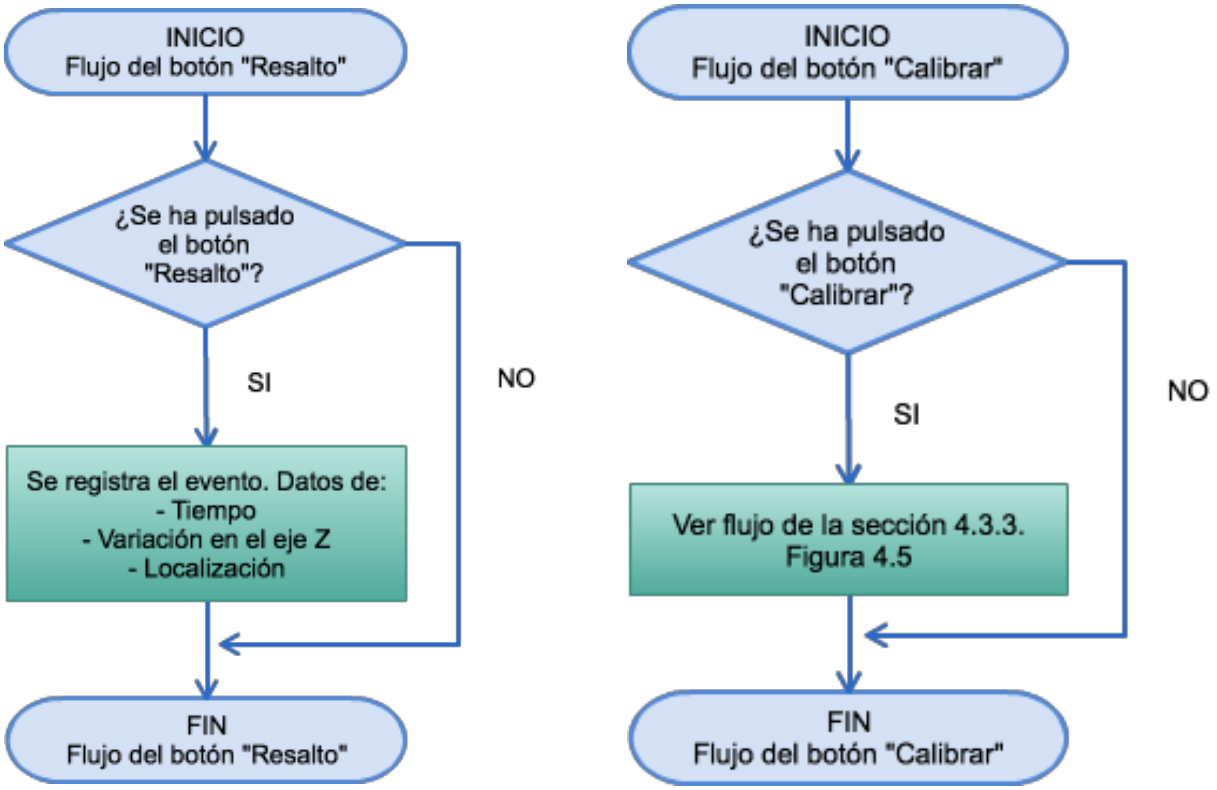


Figura 4.20. Diagrama de flujo de los botones "Resalto" y "Calibrar".

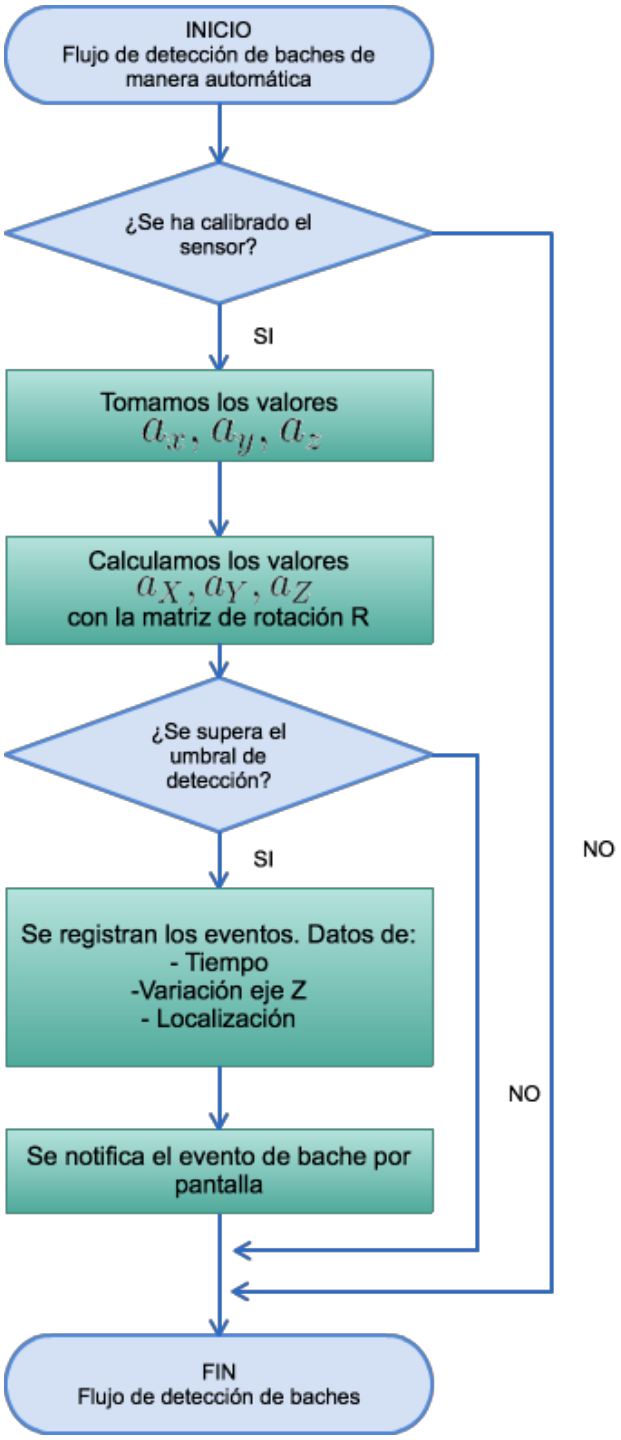


Figura 4.21. Diagrama de flujo de la detección automática de baches.

5. ENSAYOS Y ANÁLISIS DE LOS RESULTADOS

En este capítulo se expondrán los ensayos realizados como método de prueba de la aplicación, y el análisis de los resultados obtenidos. La división de las secciones en ensayos privados y públicos, tiene la intención de mostrar los diferentes ensayos efectuados.

5.1. Ensayos privados

Denominamos ensayos privados, a las pruebas ejecutadas de forma controlada por el desarrollador. En esta sección se realizan pruebas básicas, como la aparición de elementos, y pruebas más complejas que verifiquen el correcto funcionamiento.

5.1.1. Ámbito de pruebas privadas

Para la realización de los ensayos privados, ha sido necesario definir un ámbito lo más controlado posible. Por un lado, los medios para realizar las diferentes pruebas, han sido de dos tipos: virtual y físico. En concreto el entorno virtual es el proporcionado por Android Studio, y el entorno físico son los Smartphone indicados en el presupuesto (ver capítulo 6).

Por otro lado, se ha establecido un ámbito de pruebas real, que define distintos circuitos a realizar con el vehículo. Las pruebas reales iniciales se han realizado en ciudad, en zonas con circulación mínima o nula, ya que ha sido necesario depurar el software ajustando los valores de la calibración. Conjuntamente, se ha procurado que las pruebas iniciales, para comprobar la calibración, se realicen en una carretera que se aproxime lo máximo posible a una recta.

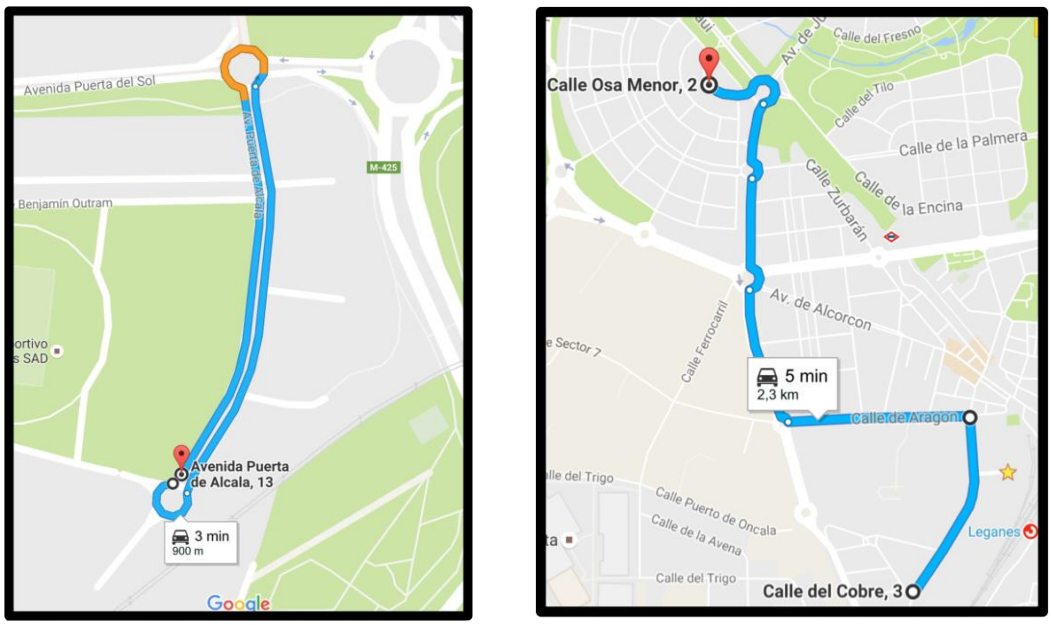


Figura 5.1. Entornos iniciales de pruebas.

Una vez comprobado el correcto funcionamiento del calibrado, se procede a comprobar el funcionamiento de la detección de baches, tanto manuales como automáticos. Para ello se escogen diferentes escenarios. En las Figuras 5.2 y 5.3 se muestran algunos de ellos.

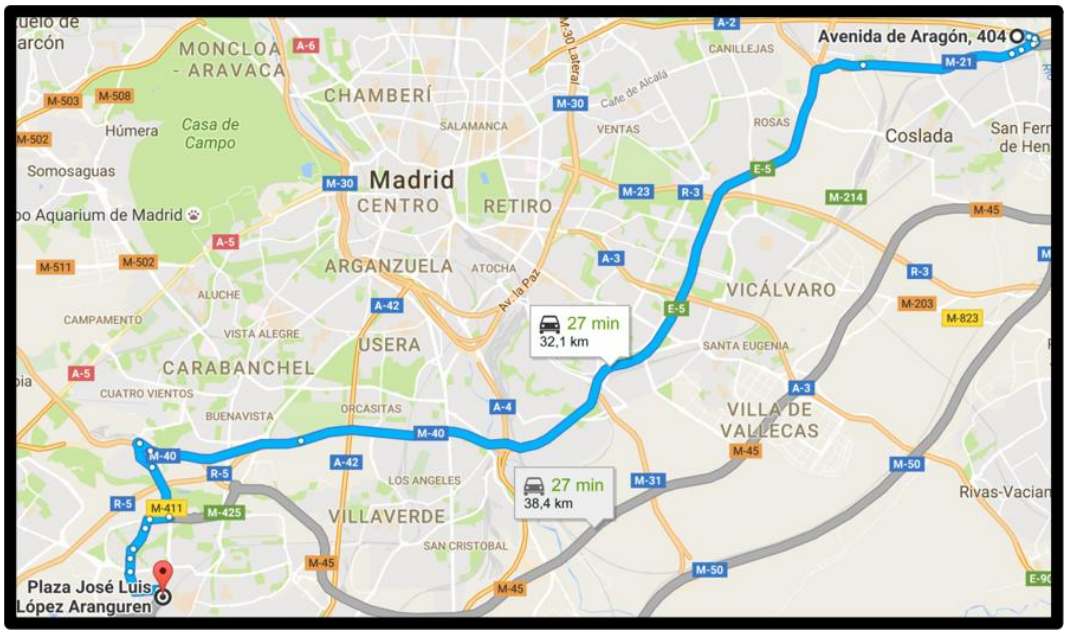


Figura 5.2. Entorno de pruebas. Carretera M - 40.

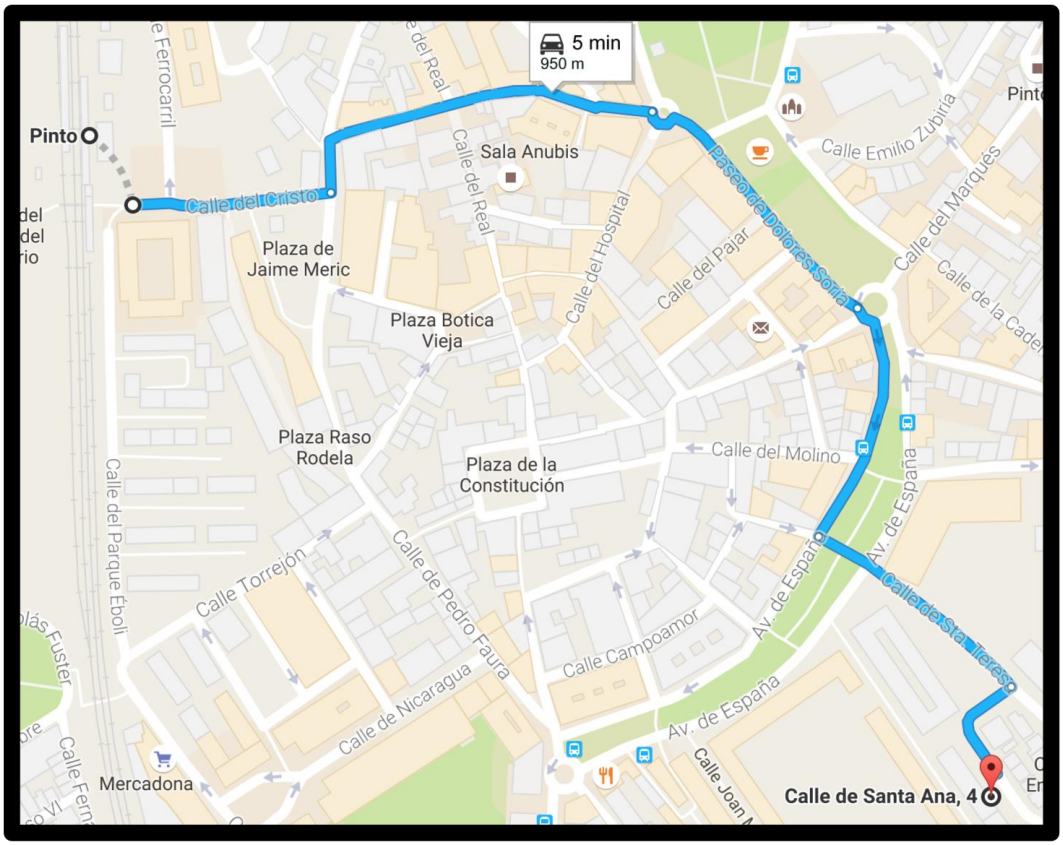


Figura 5.3. Entorno de pruebas. Carretera de ciudad.

5.1.2. Pruebas básicas de usuario

Los primeros ensayos de la aplicación, serán pruebas basadas en las diferentes tareas, e historias de usuario definidas en el capítulo 3. Se pretende demostrar así que se han cumplido todos los requisitos del usuario. Sólo se probarán aquellas tareas que afecten al funcionamiento de la aplicación final. Para ello se empleará la tabla que vemos a continuación:

ID	Descripción	Pasos	T – XX	HU – XX	Solución
Código de identificación único para cada prueba. Seguirá el código TS – XX.	Explica brevemente el propósito de la prueba.	Instrucciones a seguir para reproducir la prueba.	Código de la tarea con la que está relacionada la prueba.	Código de la historia de usuario con la que está relacionada la prueba.	Posibles valores: Éxito o fallo.

Tabla 5.1. Plantilla de pruebas.

ID	Descripción	Pasos	T – XX	HU – XX	Sol.
TS - 01	Reorientar el acelerómetro	1. Iniciar la aplicación. 2. Pulsar botón calibrado o cargar calibración del menú.	T - 04	H - 01	Éxito
TS - 02	Detección automática de baches	1. Iniciar la aplicación. 2. Calibrar el acelerómetro. 3.Verificar baches detectados automáticos	T - 05	H - 01	Éxito
TS - 03	Botón detectar baches manual	1. Iniciar la aplicación. 2. Visualizar el botón bache. 3. Pulsar el botón bache. 4.Verificar la detección manual de bache	T - 01	H - 02	Éxito
TS - 04	Verificar información sobre detección de baches en pantalla	1. Iniciar la aplicación. 2. Calibrar el acelerómetro. 3. Detectar un bache. 4. Visualizar si se muestra el evento en pantalla.	T - 02	H - 02	Éxito
TS - 05	Verificar el correcto formato de los ficheros	1. Iniciar la aplicación. 2. Calibrar el acelerómetro. 3. Detectar un bache. 4. Visualizar la información guardada en el fichero.	T - 04	H - 03	Éxito

TS - 06	Verificar que se almacena la localización de los baches	1. Iniciar la aplicación. 2. Calibrar el acelerómetro. 3. Detectar un bache. 4. Visualizar la información guardada en el fichero	T - 05	H - 03	Éxito
TS - 07	Verificar que se muestra el mensaje de calibración correcta	1. Iniciar la aplicación. 2. Calibrar el acelerómetro. 3. Visualizar la aparición del mensaje calibración correcta	T - 01	H - 04	Éxito
TS - 08	Verificar que se muestra el mensaje de calibración incorrecta	1. Iniciar la aplicación. 2. Calibrar el acelerómetro de forma incorrecta 3. Visualizar la aparición del mensaje calibración incorrecta	T - 02	H - 04	Éxito
TS - 09	Verificar la existencia de un menú de cargado y guardado de configuraciones	1. Iniciar la aplicación. 2. Verificar la existencia de ese menú	T - 03	H - 04	Éxito
TS – 10	Verificar la existencia de una opción para guardar la configuración y su funcionamiento	1. Iniciar la aplicación. 2. Calibrar el acelerómetro 3. Guardar el calibrado	T - 04	H - 04	Éxito
TS – 11	Verificar la existencia de una opción para cargar la configuración y su funcionamiento	1. Iniciar la aplicación. 2. Cargar la calibración	T - 05	H - 04	Éxito
TS – 12	Verificar que se muestran los valores del acelerómetro reorientado	1. Iniciar la aplicación. 2. Calibrar el acelerómetro 3. Verificar que se muestran los valores	T - 06	H - 04	Éxito

Tabla 5.2. Tabla de Test de Software.

5.2. Ensayos públicos

Por otro lado, se denominan ensayos públicos, a las pruebas realizadas con la colaboración de los usuarios. Para ello, se instala la aplicación en diferentes dispositivos móviles de diversos usuarios, que se presentan voluntariamente a un anuncio publicado en diversas redes sociales. Un total de 100 voluntarios prueban la aplicación y responden a un formulario para obtener su opinión.

5.2.1. Cuestionario y resultados

El cuestionario se realiza a través de la aplicación *Google Forms*, [38] desarrollada por Google, que nos permite crear un cuestionario y compartirlo con distintas personas, en el caso que nos concierne, con los usuarios de la aplicación.

El formulario creado se muestra en la Figura 5.4

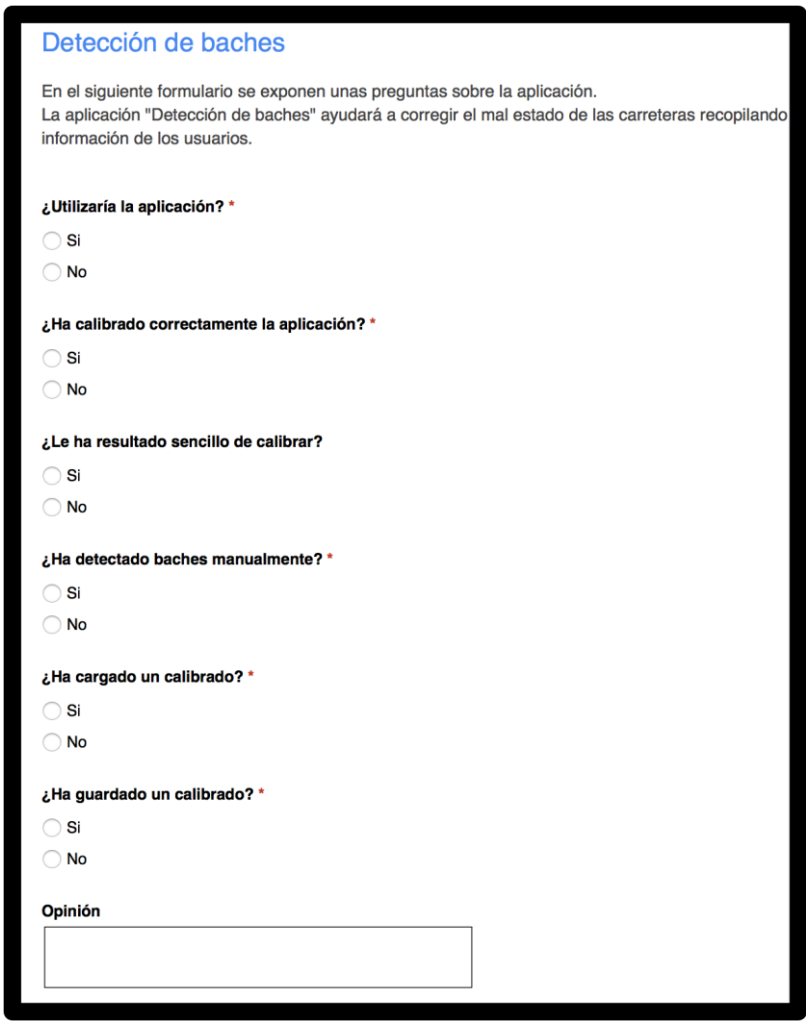


Figura 5.4. Cuestionario para los usuarios.

Procedemos a analizar las respuestas obtenidas del formulario respondido por los usuarios. Obtenemos las siguientes conclusiones:

- El 100% de los usuarios utilizaría la aplicación ya que la encuentran de gran utilidad.
- El 60% ha calibrado correctamente la aplicación, mientras que el 40% restante no lo ha hecho.
- El 80% opina que la aplicación es sencilla de calibrar. El 20% piensa lo contrario.
- Un 40% ha detectado los baches manualmente (con ayuda de un copiloto). El 60% restante ha utilizado la aplicación en modo automático.
- Mientras que un 20% ha cargado un calibrado anterior, el 80% ha preferido calibrar la APP de cero.
- Un 80% ha decidido guardar el calibrado frente al 20% que ha preferido no hacerlo.

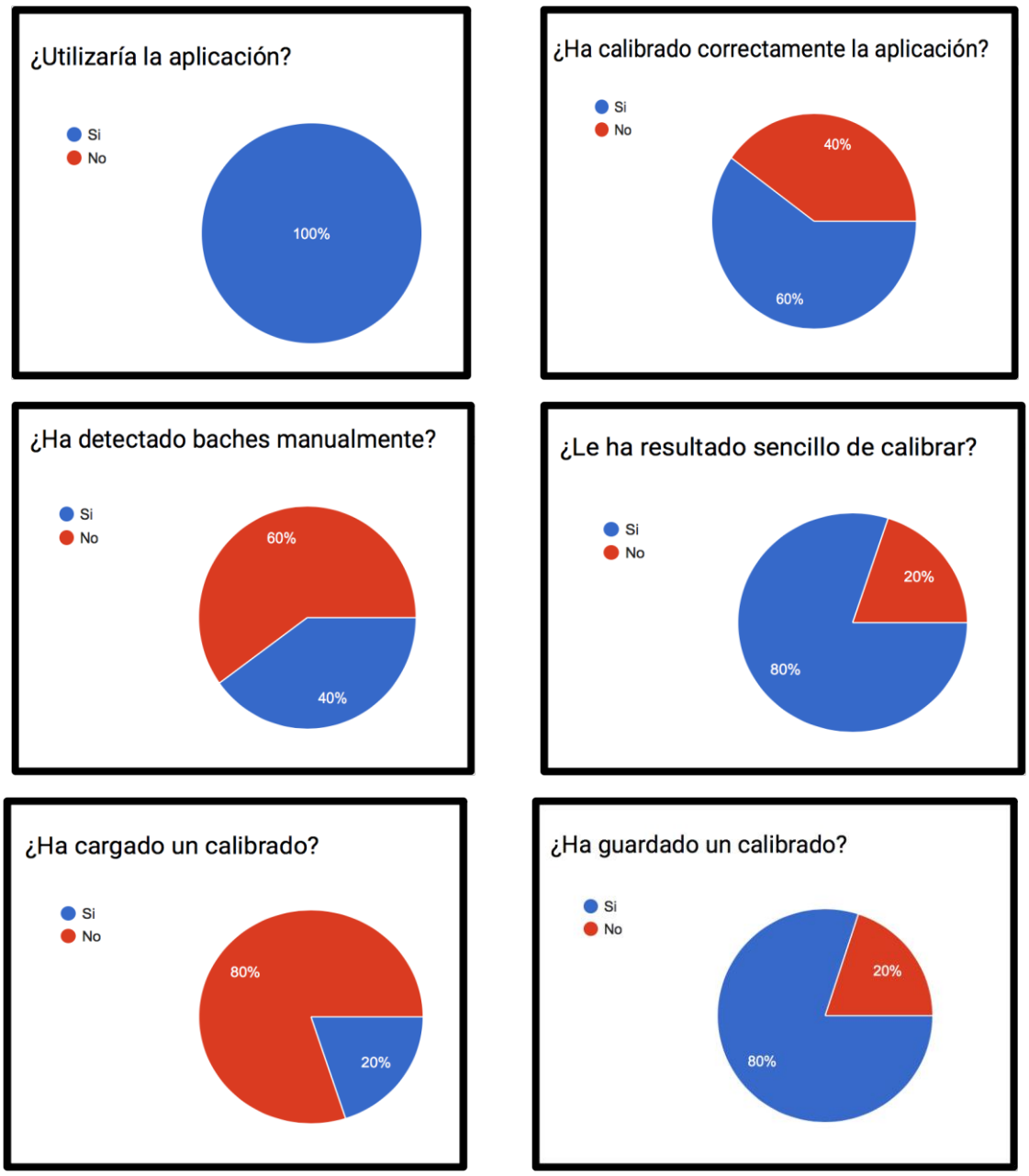


Figura 5.5. Gráficos de sectores de las respuestas obtenidas.

A continuación, se publica una muestra de las opiniones de algunos de los usuarios sobre la aplicación, así como los resultados a las preguntas:

Usuario 1: "La aplicación es muy útil, pero estaría más completa si se informara que se han detectado un número determinado de baches."

Usuario 2: "La aplicación está bien, pero le faltan matices como por ejemplo un mapa."

Usuario 3: "La aplicación cumple con su cometido. Sería interesante que además ofreciese poder navegar hacia un destino, o que funcionase en segundo plano mientras utilizo otro navegador."

Usuario 4: "Está bien, pero le faltan cosas."

Usuario 5: "La aplicación llamaría más la atención con otra interfaz gráfica, pero funciona."

	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5
¿Utilizaría la aplicación?	Sí	Sí	Sí	Sí	Sí
¿Ha calibrado correctamente la aplicación?	Sí	No	Sí	No	Sí
¿Le ha resultado sencillo de calibrar?	Sí	Sí	Sí	No	Sí
¿Ha detectado baches manualmente?	No	No	Sí	Sí	No
¿Ha cargado un calibrado?	No	Sí	No	No	No
¿Ha guardado un calibrado?	Sí	Sí	No	Sí	Sí

Tabla 5.3. Muestra de respuestas al cuestionario.

5.2.2. Análisis de las muestras

Una vez recibidos los reportes de los usuarios, es decir, los ficheros generados tras sus pruebas, se procederá al análisis de los mismos. Se escogen tres muestras al azar, y se decide representarlas con MATLAB.

A continuación, haremos un análisis de las tres muestras, y de las posibles anomalías detectadas.

Las figuras representadas a continuación muestran: En azul, la variación en el eje Z de la aceleración de forma continua. En rojo los baches detectados de forma automática. En verde los baches detectados de forma manual.

En la primera muestra, que se adjunta a continuación, observamos diversas anomalías. En primer lugar, a la izquierda del gráfico, en la zona de variación 2 y marca de tiempo 14:11:00 aproximadamente, se observa la detección de varios baches automáticos sin correspondencia manual. Con el fin de obtener una posible causa para esta anomalía, nos entrevistamos con el usuario 1. Este nos explica, que inició la aplicación y cargó la calibración antes de colocarla en el soporte, que habitualmente utiliza en el coche. Por tanto, esta es la posible causa. Por otro lado, nos indica también que tuvo problemas al informar de baches manuales, haciéndolo en ocasiones repetidas veces. Este fenómeno lo podemos observar en las proximidades de la marca de tiempo 14:11:30

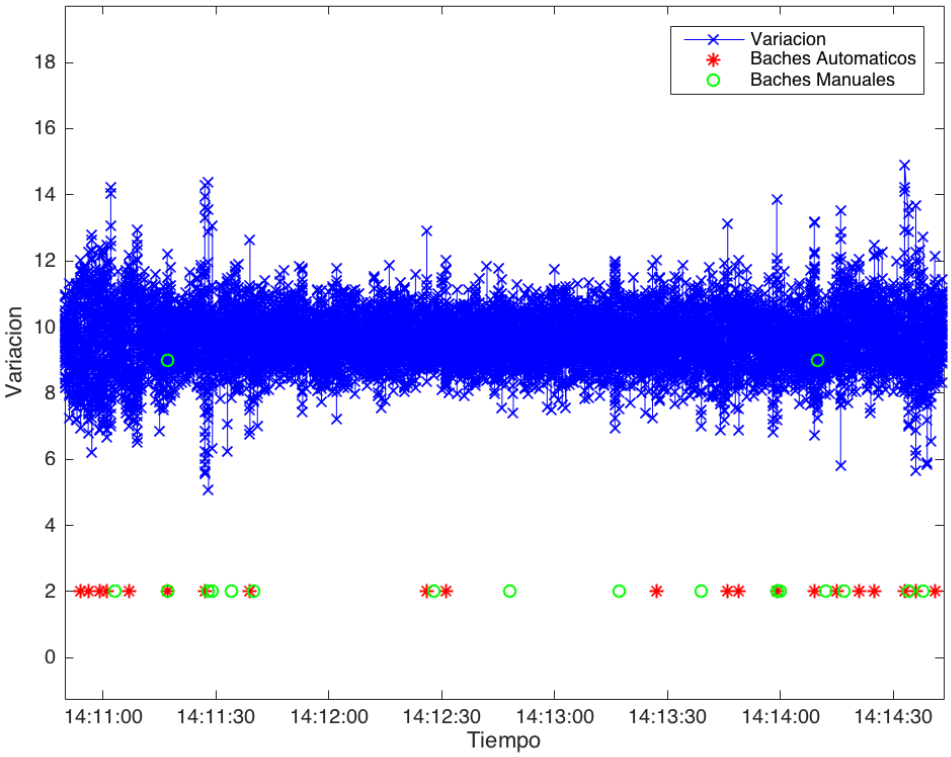


Figura 5.6. Representación de muestra del primer usuario.

En la segunda muestra, que se adjunta a continuación, observamos un pico que asombra bastante. Esta anomalía tiene lugar en las proximidades de la marca de tiempo 14:28. Decidimos comunicarnos con el usuario 2, y de la conversación obtenemos la información que explica este suceso. Durante la conducción, y captura de datos, al usuario se le

resbaló el móvil, cayendo al suelo del coche. Esto resuelve el misterio y explica el porqué de esas marcas de baches automáticos inesperadas.

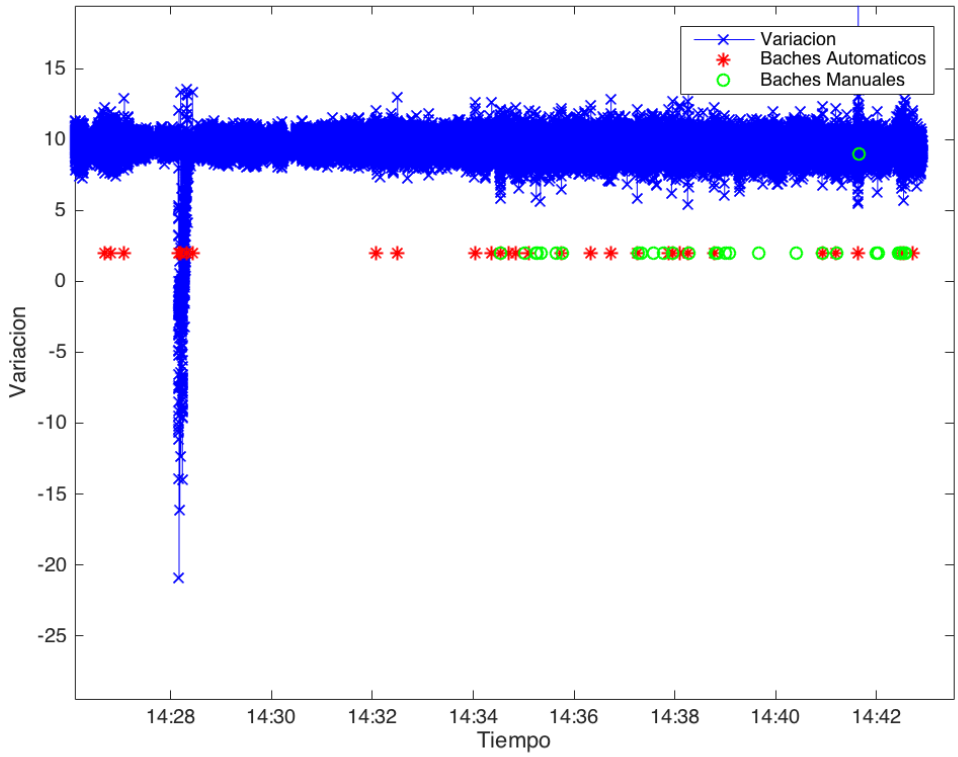


Figura 5.7. Representación de la muestra del segundo usuario.

Por último, se analiza una tercera muestra. En la Figura 5.8 se adjunta su representación. Se observa que esta representación tiene un comportamiento normal. Se detectan baches automáticos seguidos de baches manuales, lo cual es el comportamiento esperado. El sistema parece funcionar correctamente, y sin ningún tipo de incidencia a la vista.

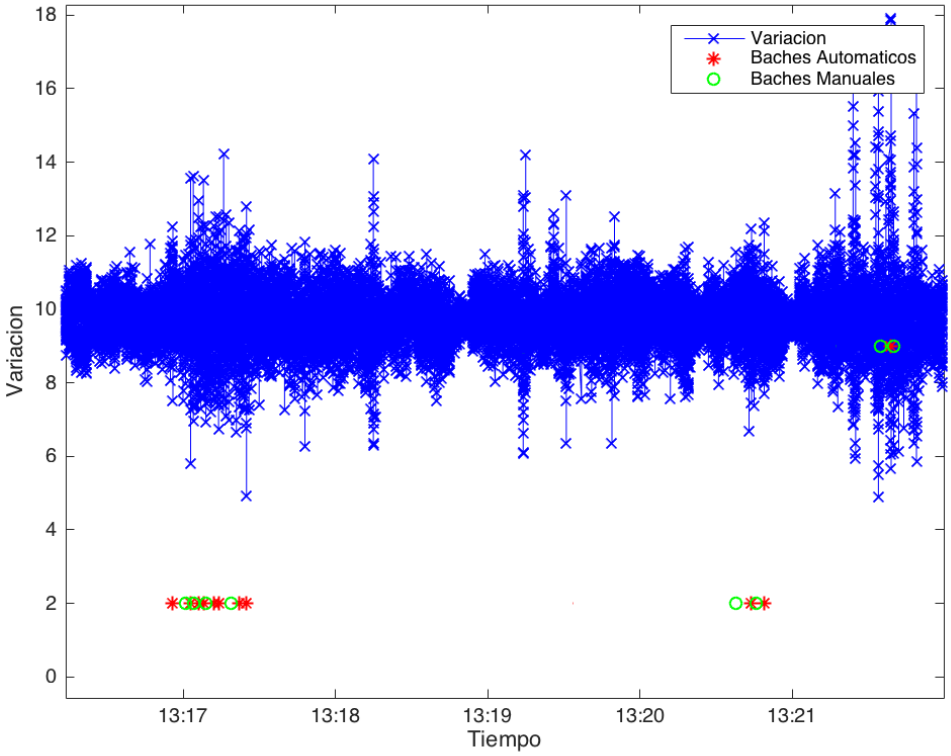


Figura 5.8. Representación de la muestra del tercer usuario.

6. GESTIÓN DEL PROYECTO

En este capítulo se expondrá la gestión del proyecto, de esta forma se podrán planificar y orientar los procesos seguidos en este TFG de manera metódica. Para ello se analizarán las etapas del mismo, añadiendo el diagrama de Gantt para explicar el tiempo previsto para las diferentes tareas. Además, se definirán los costes personales, los costes de material y los costes indirectos detallando cada uno de ellos.

6.1. Etapas del proyecto

Este apartado se ha dividido en dos secciones, por un lado, la estimación inicial de duración que se hizo al comienzo del proyecto "Aplicación Android para obtener información del estado de las carreteras". Por otro lado, la estimación final, donde se definen las etapas llevadas a cabo y el número de horas que se han dedicado finalmente.

Estimación inicial: Para realizar una estimación inicial en el tiempo, se tienen en cuenta las fechas de convocatoria para la presentación del proyecto. La universidad ofrece convocatorias en marzo, julio y octubre. Como este proyecto se inicia en noviembre, la convocatoria de enero es muy precipitada, así que la planificación inicial se realiza de cara a finalizar en julio. Para ello se realizó una estimación en horas tal y como se muestra en la Tabla 6.1.

Fases	Tiempo estimado en horas
Análisis y documentación	10 horas
Formación	30 horas
Implementación	180 horas
Pruebas y aplicación final	60 horas
Documentación	150 horas
Total	**430 horas**

Tabla 6.1. Estimación inicial del proyecto en horas.

En la Figura 6.1 se muestra el diagrama de Gantt de la estimación inicial, entre los meses de noviembre y julio.

Nombre de la tarea	Duración
Análisis del problema	28d
Lectura del artículo	8d
Investigación	6d
Ideas iniciales	9d
Formación	66d
Formación teórica Android	24d
Formación Práctica Android	61d
Implementación	75d
1. Aplicación GPS	14d
2. Aplicación acelerómetro	15d
3. Calibración	27d
4. Obtención de datos	7d
5. Algoritmo de detección	6d
Pruebas y aplicación final	11d
Integración diferentes partes	5d
Pruebas	6d
Documentación	13d
Documentación	13d

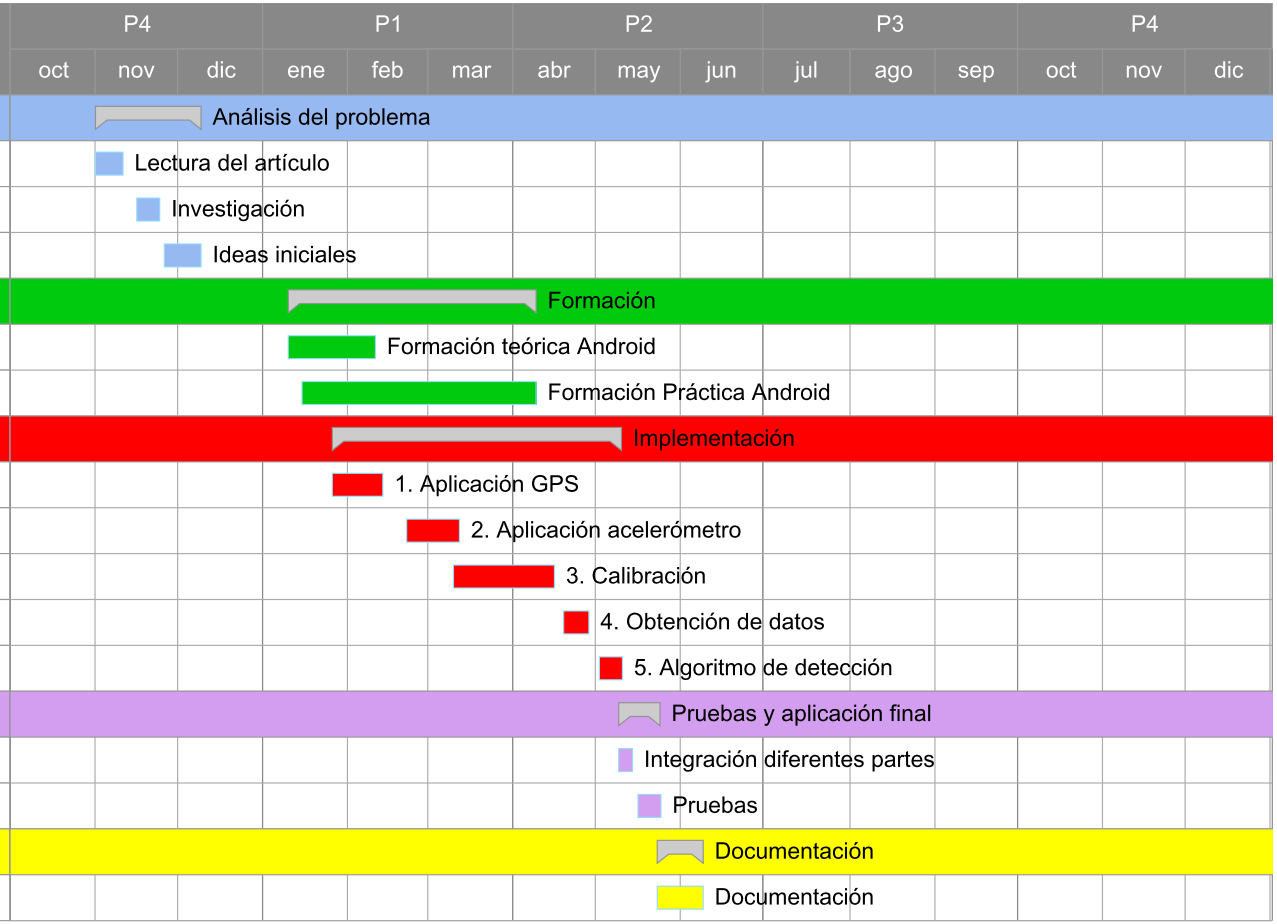


Figura 6.1. Diagrama de Gantt de la estimación inicial.

Estimación final: El desarrollo del proyecto se compagina con otras tareas universitarias: se realizan las asignaturas de 4º curso además de las prácticas de empresa. Todo esto ocasiona retrasos en el proyecto, por lo que en mayo se decide que julio no es una fecha realista de presentación y se pospone hasta octubre. Además, durante el desarrollo ha sido necesario modificar la planificación inicial, añadiendo nuevas fases. En la Tabla 6.2. se muestra la planificación final.

De las tareas expuestas a continuación, las marcadas con letra roja corresponden a aquellas realizadas de forma común con el trabajo de fin de grado "Aplicación Android para obtener información de tráfico en carreteras", Álvaro González Caballero, ya que son la base de ambos proyectos. Corresponden a un total de 130h de las 540h totales [39] (aclaración al lector).

Fases	Tiempo estimado en horas
Análisis y documentación: • Lectura del artículo • Investigación • Ideas iniciales	Total: 10 horas • 2 horas • 4 horas • 4 horas
Formación: • Formación teórica en Android • Formación práctica en Android	Total: 25 • 10 horas • 15 horas
Implementación: 1. Aplicación GPS 2. Aplicación acelerómetro 3. Calibración 4. Obtención de datos 5. Algoritmo de detección	Total: 210 horas • 20 horas • 15 horas • 60 horas • 55 horas • 60 horas
Pruebas y aplicación final: • Integración de las partes • Pruebas	Total: 75 horas • 40 horas • 35 horas
Análisis de datos: • Análisis de datos con Matlab	Total: 20 horas • 20 horas
Documentación: • Realización de la memoria	Total: 200 horas • 200 horas
Total	**540 horas**

Tabla 6.2. Duración final del proyecto en horas.

En la Figura 6.2 se muestra el diagrama de Gantt con las tareas finales y su correspondiente duración.

Además, durante el desarrollo del proyecto se han realizado reuniones en las que eran partícipes el desarrollador y el tutor. Estas reuniones han servido para realizar un seguimiento del desarrollo de este TFG así como reuniones de inicio y fin.

Se realizaron dos reuniones informativas. La primera de ellas para conocer las ofertas de TFG. La segunda para obtener información al inicio del proyecto.

Posteriormente, se realizaron 4 reuniones de seguimiento, en las cuales se han ido mostrando las fases por las que ha pasado la implementación, así como la resolución de dudas que pudieran surgir.

Por último, se realizó una reunión para la revisión del presente documento y el cierre del proyecto.

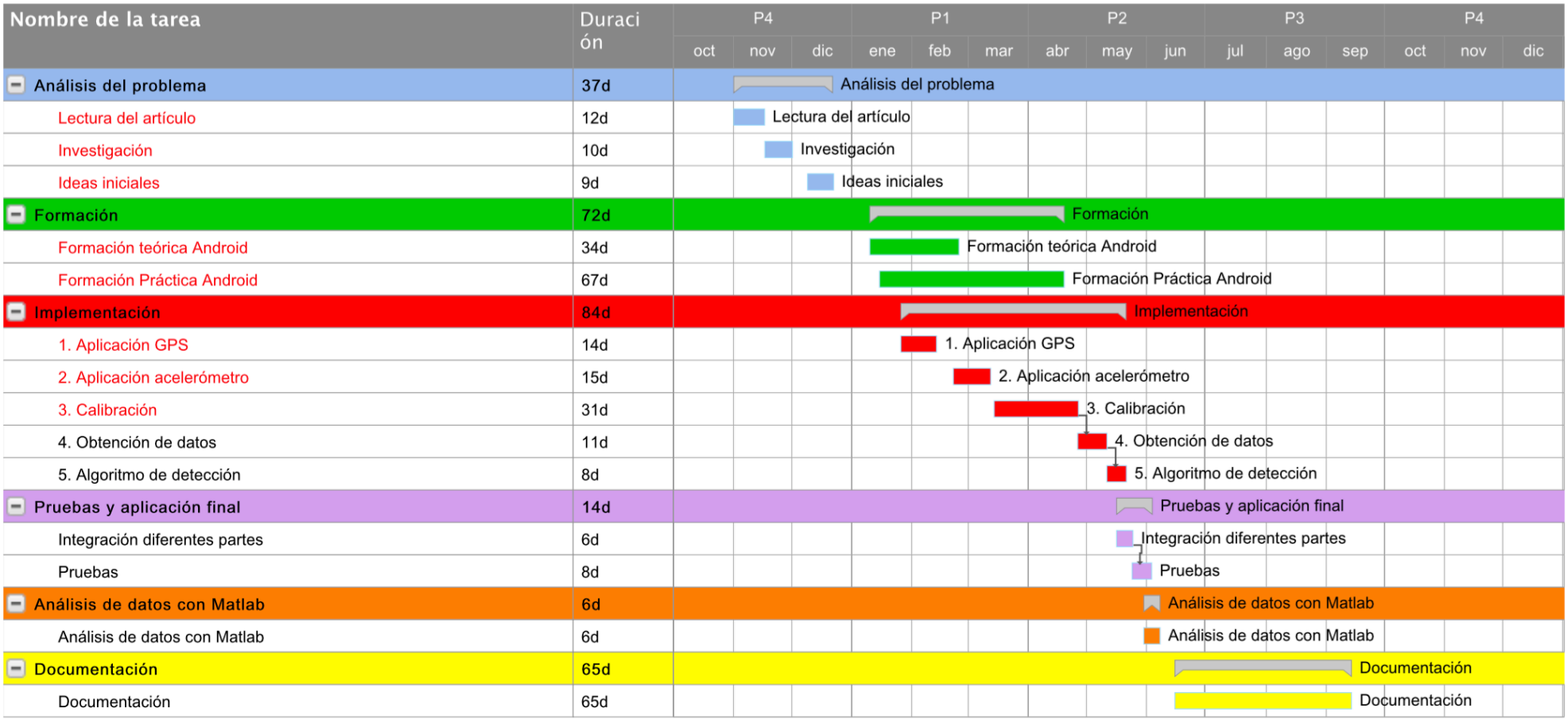


Figura 6.2. Diagrama de Gantt de la estimación final.

6.2. Presupuesto

Es requisito imprescindible en todo proyecto realizar el presupuesto del mismo. En este presupuesto se tendrán en cuenta los gastos de personal en base a las horas empleadas por cada una de las partes del proyecto, los costes materiales, hardware y software, y otros costes adicionales.

6.2.1. Coste de personal

Para calcular el coste de cada una de las personas implicadas en el proyecto, teniendo en cuenta los costes en la seguridad social, retenciones, etc. Se ha utilizado una herramienta impulsada por el BBVA, *ActiBVA* [33], cuyo objetivo es acercar la información financiera a todos los usuarios.

En este proyecto han tomado parte tres personas. Un ingeniero superior, cuyo salario medio se estima en aproximadamente 45000€ brutos anuales, y dos ingenieros junior con una experiencia menor a tres años y cuyo salario medio se estima en 25000€ brutos anuales aproximadamente.

Figura 6.3. Salarios de ingenieros junior y senior usando *ActiBVA*.

En la Figura 6.3 hemos obtenido los salarios brutos anuales tanto como para los ingenieros junior como para el ingeniero senior. En base a los datos obtenidos, calculamos el precio por hora de cada uno de ellos:

- Ingeniero senior: 23,44 €/hora brutos.
- Ingeniero junior: 13,02 €/hora brutos.

Podemos estimar el coste del personal en base a las horas dedicadas por cada parte al proyecto junto con su coste por hora y el tipo de perfil del personal.

Como vimos en la planificación, debido a aquellas tareas realizadas de forma común con el trabajo de fin de grado "Aplicación Android para obtener información de tráfico en carreteras", Álvaro González Caballero [39], se debe contabilizar al autor de este, un total de 130h en términos de presupuesto (aclaración al lector).

Personal	Perfil	Tarifa/Hora	Horas	Coste
Elsa Gómez Martínez	Ingeniera Junior	13,02€/hora	540 h	7030,80 €
Álvaro González Caballero	Ingeniero Junior	13,02€/hora	130 h	1692,60 €
Carlos García Rubio	Ingeniero Senior	23,44€/hora	50 h	1172,00 €
Total				**9895,40 €**

Tabla 6.3. Coste del personal.

6.2.2. Coste de material

Para realizar el presupuesto que supone el material, hay que tener en cuenta la vida útil de cada elemento utilizado para calcular su amortización.

El cálculo de los costes imputables se realiza según la siguiente ecuación donde:

- A: Número de meses de uso.
- B: Vida útil
- C: Coste total
- D: Porcentaje de uso dentro del proyecto

$$Coste\ imputable = \frac{A}{B} \cdot C \cdot D$$

Ecuación 6.1. Ecuación del coste imputable.

	Producto	Precio (€)	Uso	Vida útil	% de uso	Coste imputable (€)
HW	Smartphone BQ Aquaris A4.5	178,04 €	10 meses	36 meses	100 %	49,45 €
	Smartphone BQ Aquaris E5	159,00 €	1 mes	36 meses	30 %	1,33 €
	Ultrabook Macbook Pro Retina 13’	1649,00 €	10 meses	60 meses	100 %	274,83 €
	Ultrabook Asus UX32VD 13’	999,99 €	3 meses	48 meses	25 %	15,62 €
SW	Android Studio	0 €	10 meses	-	100 %	0 €
	Licencia Android	0 €	10 meses	-	100 %	0 €
	Licencia OS X	0 €	10 meses	-	100 %	0 €
	Paquete ofimática Microsoft Office	79,00 €	4 meses	48 meses	50 %	3,29 €
	Licencia Matlab	69,00 €	2 meses	-	30%	41,40 €
	Total					**385,92 €**

Tabla 6.4. Coste del material. Hardware y software.

6.2.3. Costes indirectos

Por último, es necesario incluir al presupuesto otros costes imputables tales como el combustible del vehículo, el propio vehículo, o las tarifas de luz e Internet. Puesto que es difícil cuantificar todo lo relacionado con estos costes, se ha decidido utilizar unos valores medios.

Producto	Coste (€)	Uso	Vida útil	% de uso	Coste imputable (€)
Tarifa de Internet	50 €	10 meses	-	50 %	250 €
Tarifa de luz	210 €	10 meses	-	50 %	1050 €
Vehículo	0,35 €/km	2000 km	-	-	700 €
Total					**2000 €**

Tabla 6.5. Otros costes.

6.2.4. Coste total

Una vez obtenidos los costes por grupos, podemos obtener el coste total del proyecto.

Tipo de coste	Coste
Costes de personal	9895,40 €
Costes de material hardware	385,92 €
Otros costes	2000,00 €
Total	**12281,32 €**

Tabla 6.6. Coste total.

7. CONCLUSIONES Y FUTURAS MEJORAS

En este capítulo se detallarán las conclusiones obtenidas con la realización de este proyecto. Además, se expondrán líneas de trabajo futuras que mejorarán la aplicación.

7.1. Conclusiones

Las conclusiones de esta memoria, no sólo afectan al cumplimiento de los objetivos marcados. También afectan a lo aprendido académicamente durante el desarrollo del proyecto.

Tras las pruebas realizadas, concluimos que el desarrollo de este trabajo ha sido un éxito. Pero para confirmarlo, he decidido realizar un repaso a los objetivos establecidos al comienzo del proyecto.

Por un lado se buscaba una solución que ahorrase en coste, y fuese eficaz. Esto se ha conseguido utilizando algo al alcance de millones de personas: un Smartphone. La solución además debía ser fácil de usar, y para ello se ha utilizado la plataforma Android, que como pudimos ver en el estado del arte, es sin duda la mejor de las opciones para un proyecto de este tipo. Se debía proteger al usuario así como su privacidad, algo que he conseguido gracias a reportar únicamente los eventos necesarios. Además, se buscaba la detección de los desperfectos en el pavimento de manera automática. Esto ha sido posible gracias a la creación de un algoritmo propio que ha sido implementado en el Smartphone.

El último de los objetivos a cumplir es ayudar a reducir los accidentes. Aunque el impacto de esta aplicación en la disminución del número de accidentes de tráfico es difícil de cuantificar, este objetivo es sin duda mi mayor motivación. El poder contribuir a ello, aportando una solución al alcance de todo el mundo, supone un logro personal. Tal vez este proyecto siembre las bases de futuras soluciones utilizadas en el mundo real. Esto será sin duda el mayor objetivo que se pueda cumplir.

7.2. Futuras mejoras

A pesar de haber cumplido los objetivos propuestos, mientras se desarrollaba la aplicación, han ido surgiendo propuestas de mejora para un funcionamiento más completo. Además, se han tenido en cuenta las opiniones de los usuarios tomadas del formulario con el fin de conseguir una aplicación más al gusto del consumidor. Algunas de las futuras mejoras son:

- Mostrar en la pantalla el número de baches detectados durante el funcionamiento de la aplicación.
- La automatización del calibrado sin necesidad de interactuar con el usuario.

- Implementar el envío de datos a través de las redes de comunicaciones para que los usuarios puedan comunicar sus baches detectados.
- Implementar un navegador en la aplicación o conseguir que ésta sea capaz de funcionar en segundo plano para que los usuarios puedan usar otras aplicaciones.
- Desarrollar una funcionalidad nueva que permita la detección de curvas demasiado pronunciadas de manera automática.
- Desarrollar la aplicación en los diferentes sistemas operativos móviles, con preferencia en iOS dado su mayor volumen de mercado respecto al resto.
- Programa de incentivos para motivar al usuario a utilizar la aplicación.

ANEXO A – MANUAL DE USUARIO

Este anexo sirve como breve guía para el usuario. En él se explican los pasos a seguir para utilizar la aplicación.

Tras instalar la aplicación, nos situaremos en la pantalla principal de nuestro Smartphone, pulsaremos el botón Menú, y buscaremos el icono de nuestra aplicación.

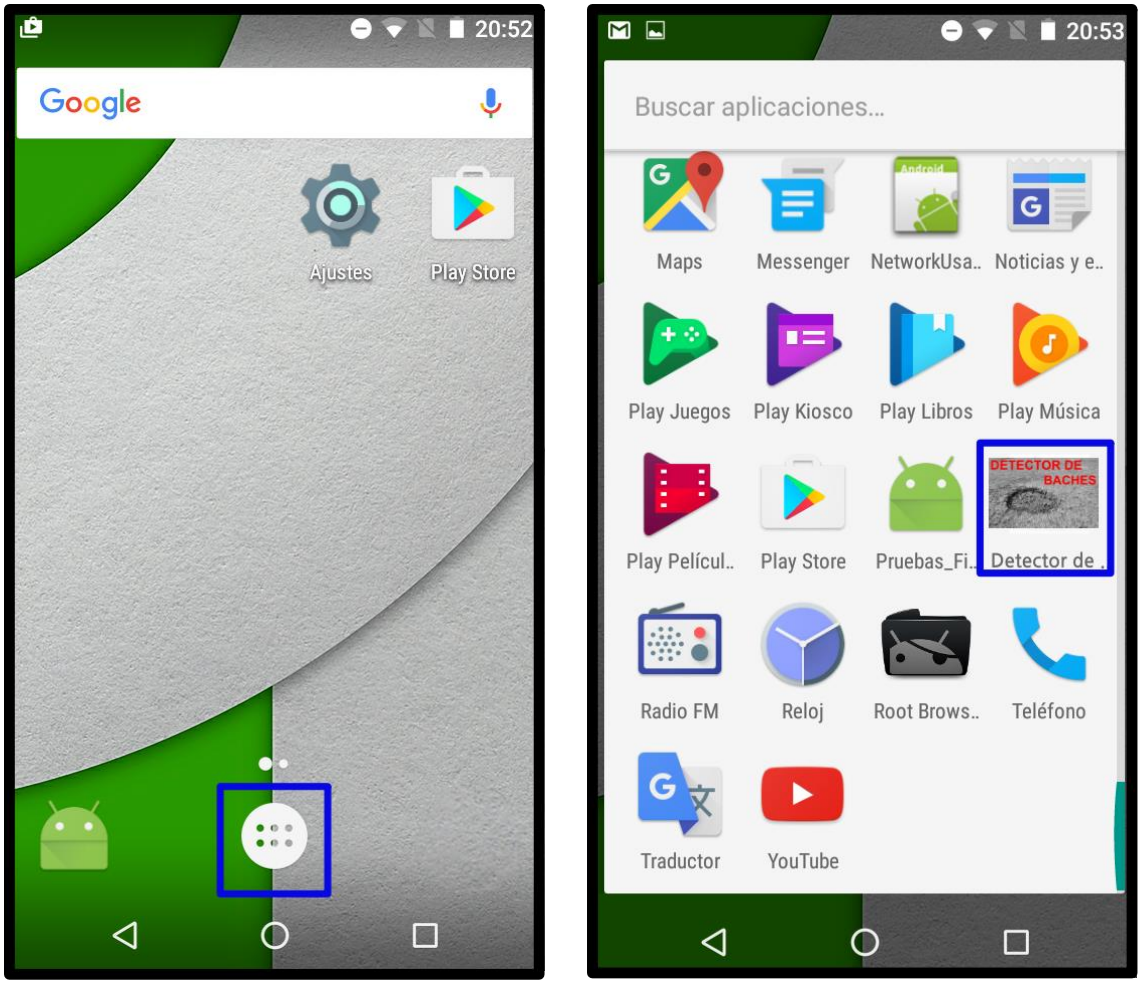


Figura A.1. Pantalla principal y menú desplegable

Al iniciar la aplicación se nos solicitará conceder los permisos necesarios para el uso de la misma. Estos son: Permisos de almacenamiento y permisos de ubicación. Es necesario aceptar ambos, de otra forma, la aplicación no funcionará.

Cuando los permisos sean aceptados, no volverá a aparecer ese mensaje a no ser que el usuario los vuelva a restringir.

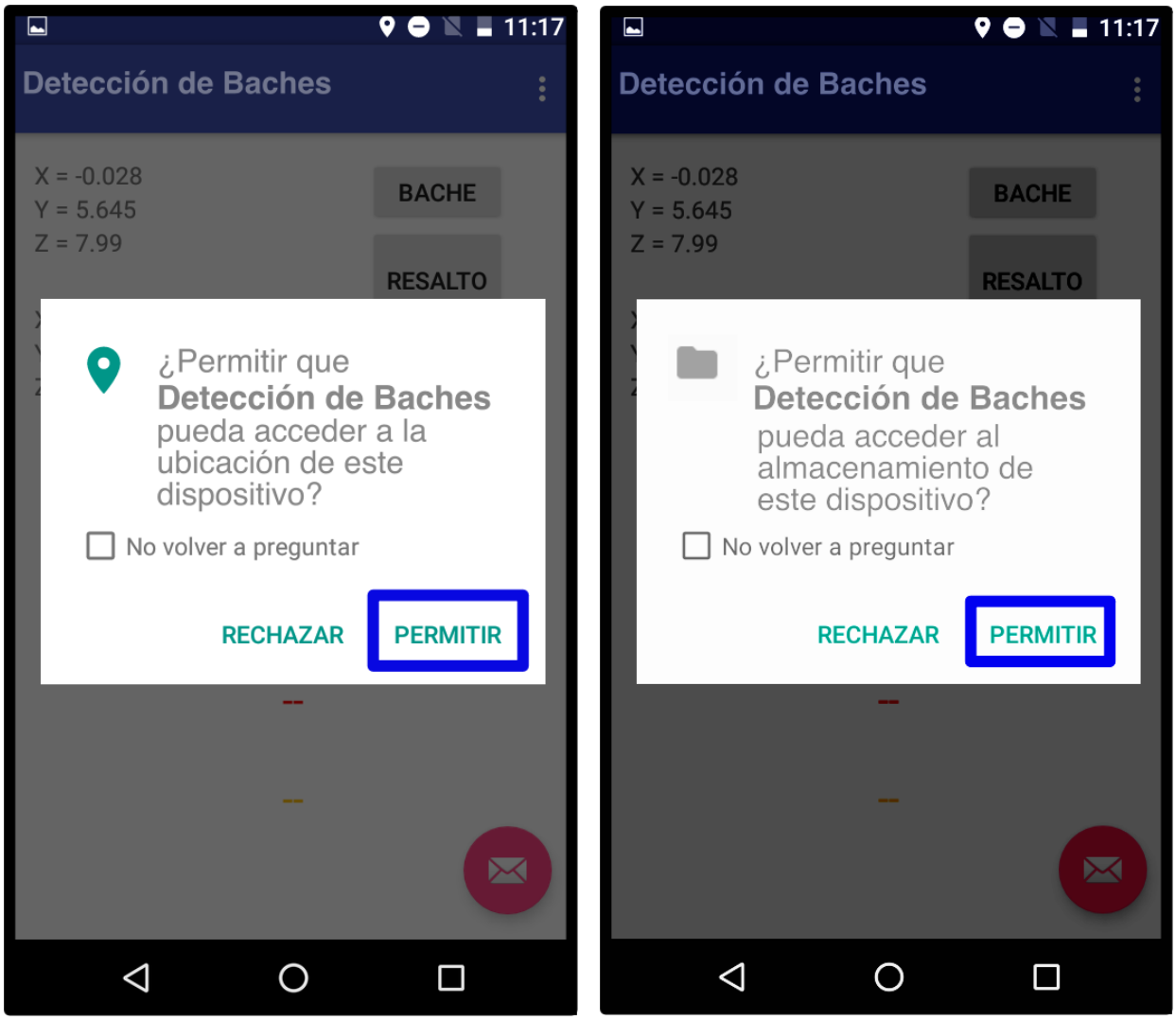


Figura A.2. Solicitud de permisos al usuario.

Una vez aceptados los permisos, nos encontramos con la interfaz de la Figura A.3. Dicha interfaz consta de los ejes del valor del acelerómetro, un botón de bache, un botón de resalto, un botón de calibración, un menú desplegable e información por pantalla.

Los elementos de esta interfaz son:

- **Aceleración del vehículo:** Nos permite controlar los ejes de aceleración del vehículo. Es información para el usuario.
- **Botón "Calibrar":** Su función es comenzar la calibración del dispositivo. Permite avanzar en las fases de ésta.
- **Información general de la aplicación:** Aquí se mostrarán los pasos a seguir de la calibración, si se detecta un bache o una frenada.
- **Menú de configuraciones anteriores:** Este menú desplegable nos dará la opción de guardar un calibrado o cargar uno anterior.
- **Botón "Bache":** La función de este botón es almacenar un evento de bache manualmente.

- **Botón "Resalto":** La función de este botón es almacenar un evento de badén manualmente.

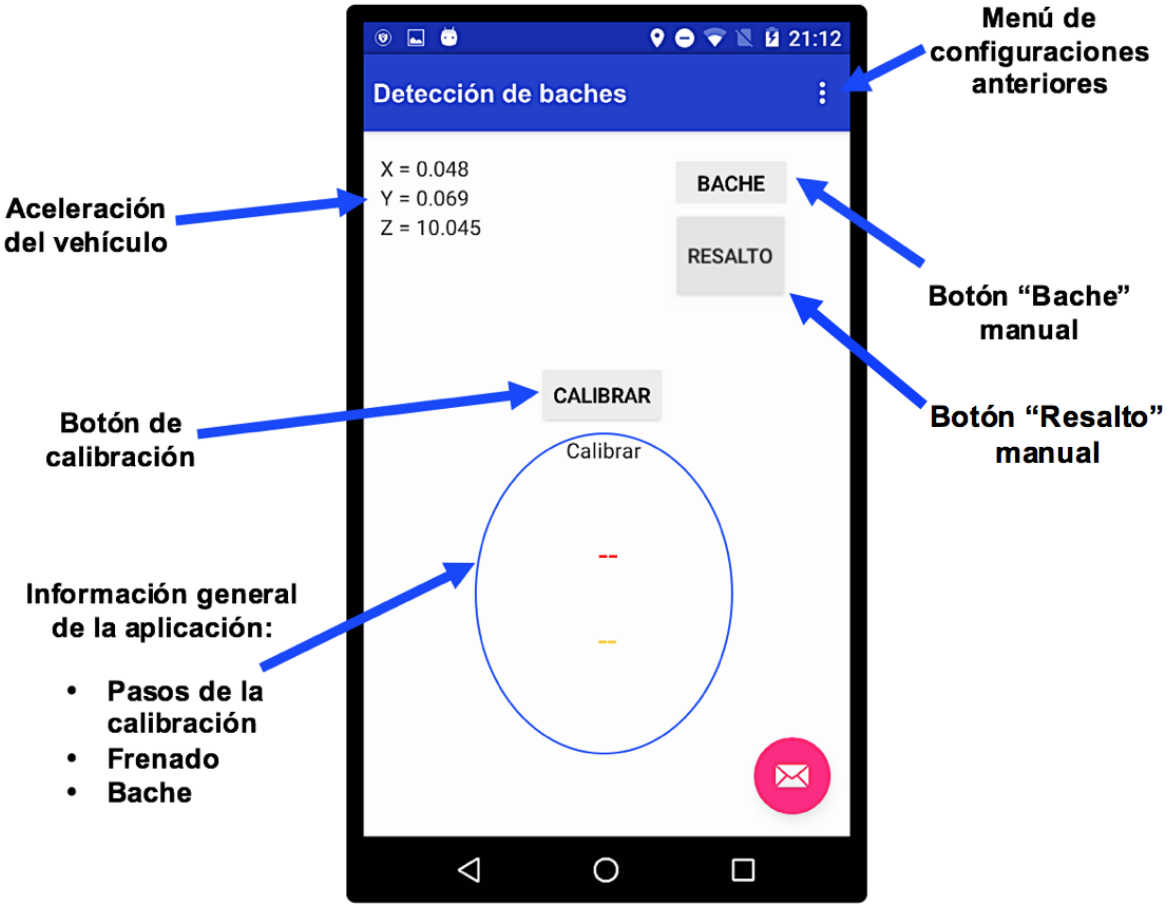


Figura A.3. Vista principal de la aplicación.

Una vez conocidos todos los elementos de la pantalla principal, comenzaremos a usar la aplicación.

1. Calibrar el dispositivo

Si ya se ha calibrado el dispositivo anteriormente y se ha almacenado la configuración, ir al apartado 3 del manual. Si no, seguir los pasos que se detallan a continuación.

Antes de comenzar, es necesario colocar el dispositivo en una posición fija. Deberá permanecer en la misma posición hasta su próximo calibrado o el cierre de la aplicación. La calibración consta de dos fases: fase de reposo y fase en movimiento.

1.1. Fase de calibración en reposo

Esta fase comienza tras pulsar el botón “Calibrar”. En la zona de información general de la aplicación, se mostrará el siguiente mensaje “Mantenga el vehículo parado” y comenzará una cuenta atrás durante 5 segundos. Es preciso que el terminal permanezca en reposo durante esta fase.

Figura A.4. Fase en reposo.

Cuando la cuenta atrás finalice, podremos pasar a la fase de calibración en movimiento.

1.2. Fase de calibración en movimiento:

Tras finalizar la fase de calibración en reposo, podremos observar que aparece un nuevo mensaje en la pantalla. “Fase en reposo finalizada. Pulse el botón “Calibrar” e inicie la marcha”.

Una vez el usuario haya pulsado el botón “Calibrar”, comenzará la fase de calibración en movimiento. El usuario únicamente tendrá que iniciar la marcha ya que la aplicación tomará de manera automática los datos necesarios. Cuando la aplicación haya tomado los datos, aparecerá un nuevo mensaje “Calibración finalizada con éxito” en la interfaz de usuario.

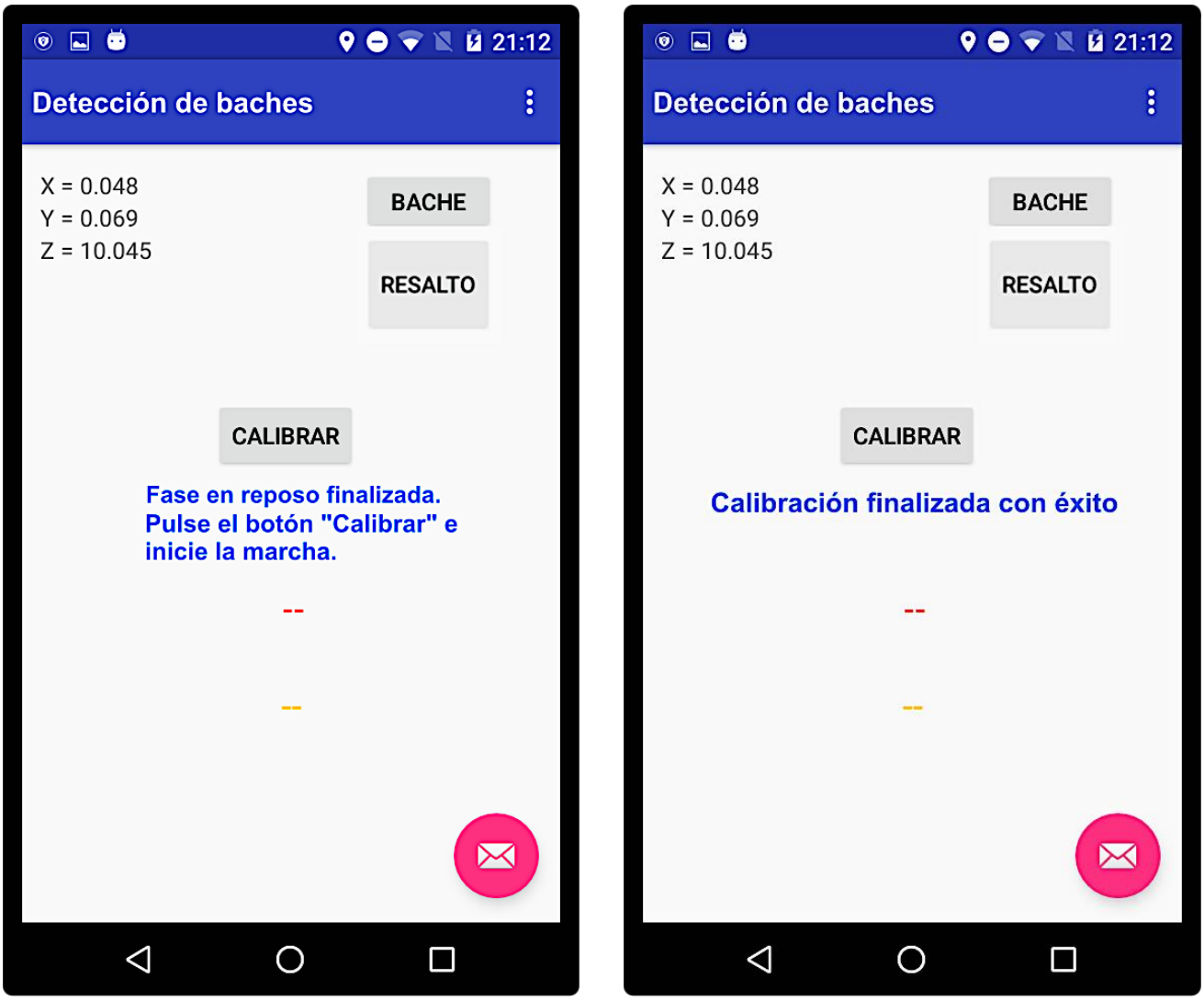


Figura A.5. Fase en movimiento.

Con la aplicación calibrada, podemos utilizarla correctamente. Podremos observar que en la información general para el usuario se muestran los mensajes “Frenando”, en amarillo, y “Bache!!”, en rojo, cuando se produce una frenada o un bache respectivamente.

2. Obtención de datos

Tras finalizar la calibración, iniciaremos la fase de toma de datos. Existen dos formas: toma de datos de manera manual y la toma de datos de manera automática. Es importante mencionar que la toma de datos de manera manual solo debe realizarla una persona que no vaya conduciendo el vehículo, ya que de lo contrario estaría infringiendo la ley.

2.1. Toma de datos manual

Cuando el usuario pulse el botón “Bache” o el botón “Resalto” de manera manual, la aplicación almacenará en un fichero la posición de los eventos registrados para su posterior análisis.

2.2. Toma de datos automática

Sin necesidad de que el usuario pulse el botón “Bache” o el botón “Resalto” de manera manual, la aplicación almacenará en un fichero la posición de los eventos registrados para su posterior análisis.

3. Cargar/Guardar calibrado el dispositivo

En este apartado se explicarán las opciones de guardar un calibrado que hemos realizado o cargar uno que ha sido guardado previamente.

3.1. Guardar un calibrado

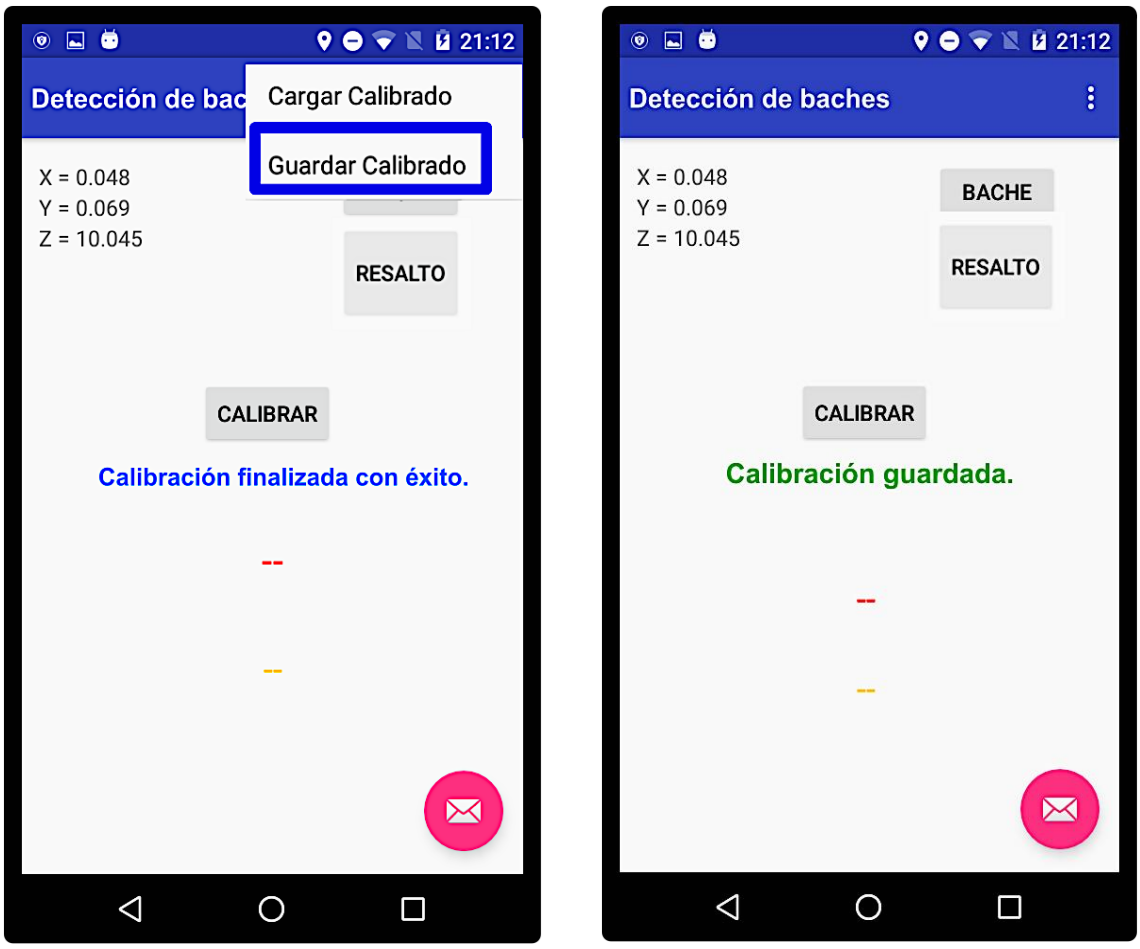


Figura A.6. Guardar calibrado.

Si hemos realizado una calibración y queremos guardarla para usos posteriores, accederemos al menú de configuraciones anteriores, y

cuando el menú esté desplegado, seleccionaremos la opción de “Guardar Calibrado”. La calibración se guardará en un fichero de texto. Y aparecerá el mensaje “Calibración guardada”. Ver Figura A.6 para más detalles.

3.2. Cargar un calibrado

Si hemos guardado una calibración previa y queremos cargarla, accederemos al menú de configuraciones anteriores, y cuando el menú esté desplegado, seleccionaremos la opción de “Cargar Calibrado”. La calibración se cargará. En la pantalla aparecerá el mensaje “Calibración cargada” y ya podremos empezar a utilizar la aplicación como en la fase 2. Ver Figura A.7 para más detalles.

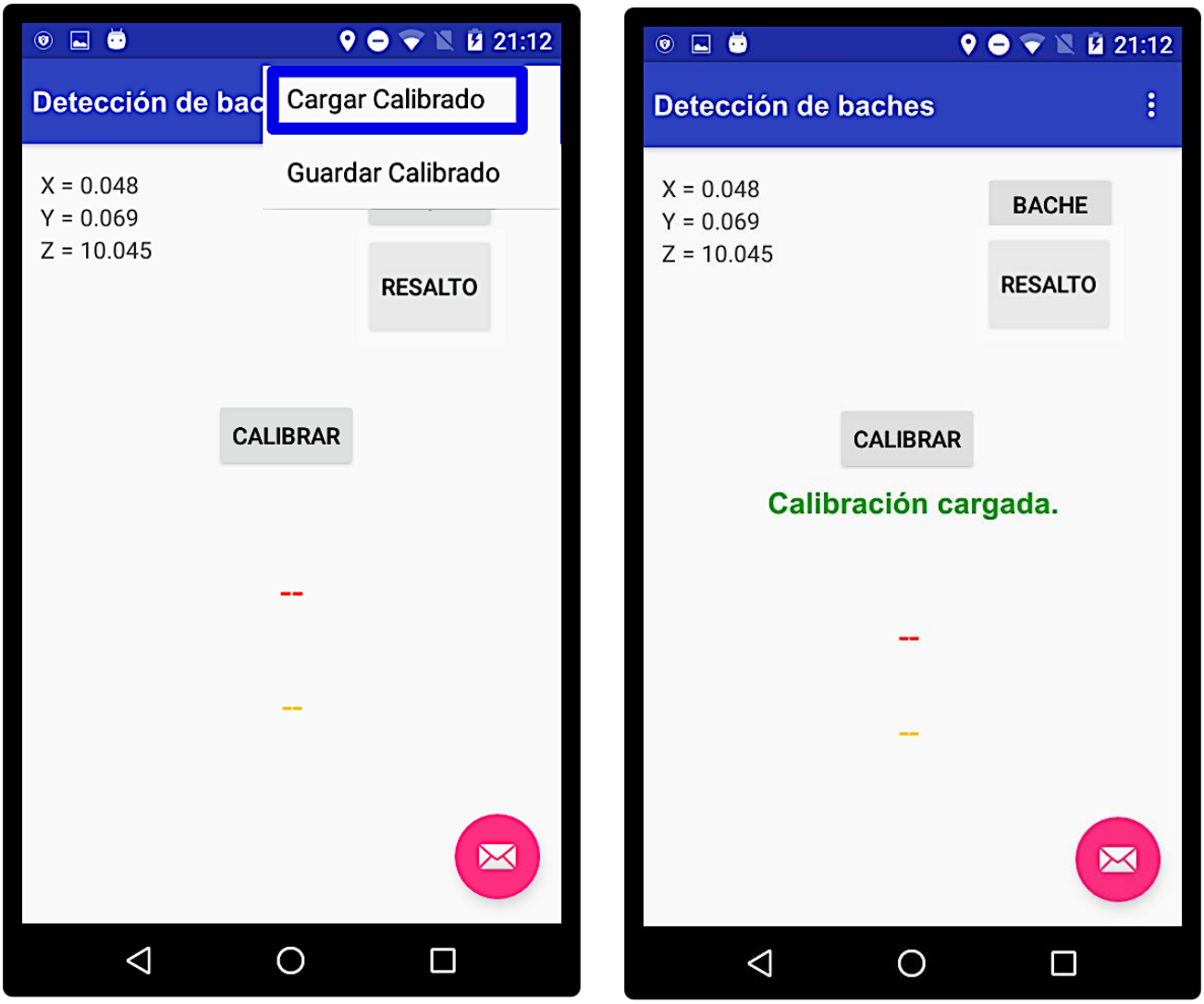


Figura A.7. Cargar calibrado.

ANEXO B – GLOSARIO

En este capítulo se expone una lista con las definiciones de las siglas, abreviaturas o palabras extranjeras utilizadas durante el presente documento.

- **AEA:** Automovilistas Europeos Asociados.
- **API:** Interfaz de Programación de Aplicaciones (*Application Programming Interface).*
- **APP:** Abreviatura de aplicación.
- **ART:** Android Runtime.
- **BBVA:** Banco Bilbao Vizcaya Argentaria.
- **CEA:** Comisariado Europeo del Automóvil.
- **CPU:** Unidad Central de Procesamiento (*Central Procession Unit).*
- **DGT:** Dirección General de Tráfico.
- **EOD:** Trabajo de Fin de Grado (*End-Of-Degree).*
- **IDE:** Entorno de Desarrollo Integrado (*Integrated Development Environment).*
- **iOS:** Sistema operativo Iphone (*Iphone Operating Sistem*).
- **RAE:** Real Academia Española.
- **ROM:** Memoria de solo lectura (*Read-Only Memory).*
- **SO:** Sistema operativo.
- **TFG:** Trabajo de Fin de Grado.
- **UML:** Lenguaje Unificado Modelado (*Unified Modeling Language).*

ENGLISH VERSION

1. INTRODUCTION AND OBJECTIVES

In this chapter, I will make a brief introduction to my project as well as the reason why I chose this topic and the objectives to be achieved. Besides, I will explain the different developmental stages of the project, the resources used and the structure of this document.

1.1. Introduction

The world of telephony has enormously improved since the first Smartphone in 1994 [1] in such a way that we moved from using it for limited functions to integrate it into our daily routines. This fact allowed smart phones to be developed with greater capacity and performance. This, together with different platforms throughout, this time, has allowed to the emergence of unlimited mobile applications that meet all sorts of requirements that a user of this kind of gadget may have.

It is also true that the use of the automobile increases every year as it can be seen in the Average Daily Traffic (ADT) of the Autonomous Region of Madrid [2]. Traffic has increases 2,64% when compared with the results in 2014 and this survey has also mentioned that the vehicle fleet in the Community of Madrid has increased 0,19% with a growing vehicle population of 4.200.832.

These two elements, telephony and vehicles, are key issues in the development of the project aimed at users of both things.

The Spanish Road Association (AEC) highlights that the quality of the road demands an investment of 6.617 million Euros in order to be in “appropriate” conditions. The survey made by AEC analyses 3.000 road sections in Spain in which only the ones in the Basque country and Extremadura are considered to be in “acceptable” conditions. [3]

According to the Directorate General of Traffic (DGT), 10% of car accidents are due to poor road conditions [4][5]. On the other hand, CEA Foundation (European Automobile Commissioner) has referred to the investment in conservation of Spanish roads as insufficient. Only 33% of State roads and 45% of regional roads can be considered to be “good”. These AEA (European Drivers Association) data explain that this is consequence of the 40% budget cut of the Ministry of Development aimed at the conservation of roads between 2008 and 2012. [6]

The bad quality of the road causes traffic accidents, at times mortal, apart from the damage in cars (tyres, shock absorbers, steering control, etc.)

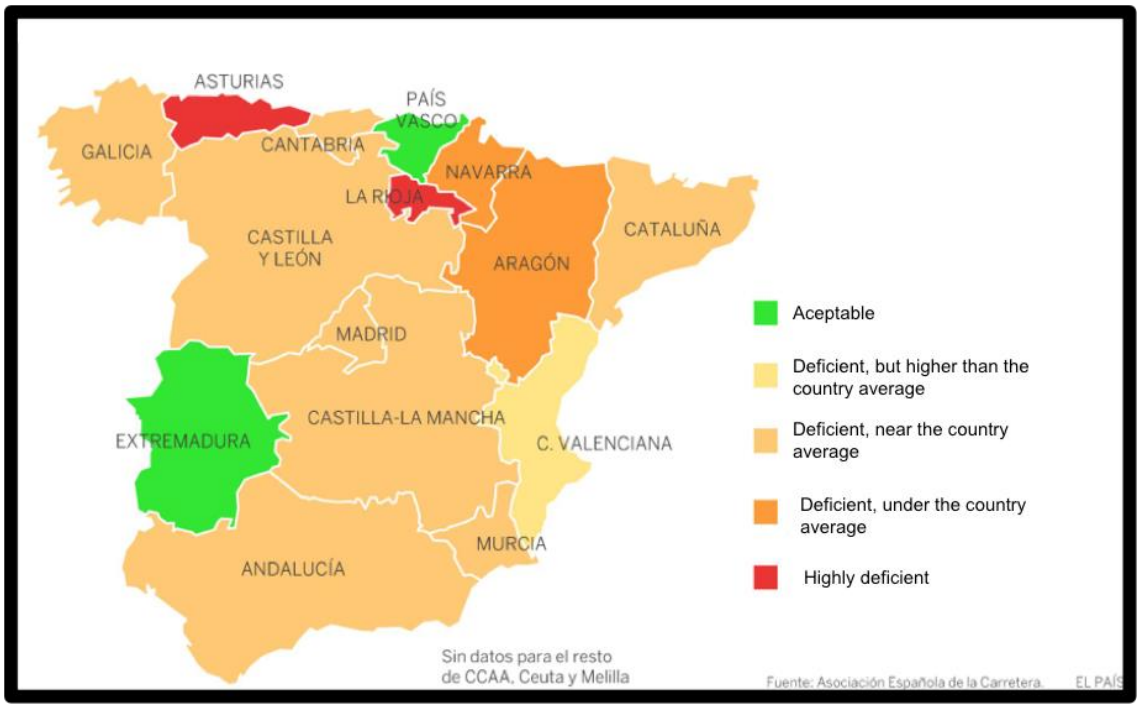


Figure C.1. Pavement Surface condition of the roads in Spain. [3]

The different solutions available are considered inefficient or highly expensive. On the one hand, DGT with its maintenance vehicles, reports the status of traffic signals or road condition manually, which means a poorly effective and inaccurate method.

On the other hand, vehicle manufacturers like *Land Rover* and *Jaguar* have developed a system called *MagneRide,* which allows detecting the condition of roads automatically due to its specialised sensors. These sensors must be really precise, turning out to be very costly and making this solution almost unaffordable economically speaking [7]. We also discovered an initiative by Bose which wanted to keep the movement of the body close to zero. It was finally dismissed because car manufacturers considered it as heavy and expensive. [8]

Jointly, we can find prevention campaigns like Ponle Freno [9], CEA Foundation [10] or AEA [11] which claim for reporting this situation finding cracks, bumps and holes on the road in order to minimize the amount of road accidents.

This project tries to find out the bumps on the road and therefore reduce the number of accident due to this situation. Because of technological breakthroughs over the last years, the raise of the Smartphone use and the increase in the amount of cars on the road, they decided to create a new application able to automatically detect a bump and record its geographical location by using the sensors directly from the Mobile device like the GPS and accelerometer.

Al present, there are several mobile devices but two of them are in leading position: the operating system Android (Android So, from now

onwards) and iOS. The platform chosen for the development of this Project is Android because it enables access to a large number of users among other advantages that will be analysed in chapter 2.

1.2. Objectives

The End-of-Degree Project detects irregularities on the roads like bumps. The solution must be an alternative to the ones already used, making the development of a new solution possible. It has to be cheap, easy to use, protecting the user, and preventing him or her from serious injuries (in potential accidents caused by bumps). Down below we explain these objectives in depth.

- **Cost savings:** This Project must develop a cheap solution that could be used by the general public. For this reason, we are using mobile devices and Android.

- **Easy to Use:** The solution must be easy to use, that's why we will implement an automatic mechanism to detect the bumps.

- **Protection to the user:** The respect for the user's privacy and the resource savings in his or her device (like the battery), would be the key to achieve the expansion of the application in order to have numerous data sources.

- **Reduction of Accidents:** If the objectives already mentioned above are fulfilled, the number of accidents could be diminished compared to previous statistics. Lives can be saved if we all collaborate in this.

Thus, we can sum up the main objective as to develop an application open to the general public (due to its low cost and the minimum resources need) that allows the detection of bumps and therefore, avoid accidents (improving the road conditions).

1.3. Project development stages

This project has been divided into different stages that are detailed below. It is important to clarify that the stages of the documentation, training and a part of the implementation stage, have been carried out in a common way with the end of grade work "Android application to get traffic information on the road", Álvaro González Caballero [39] (clarification to the reader).

Stage 1 – Documentation: During the documentation stage, we proceeded to study the problem, looking for the best way to detect poor road surface. Some of the requirements of the application, among many others, are the need of a prior calibration in the Smartphone, or that the terminal has sensors required by the application. Also, to give alternative solutions that meet the objectives detailed in the previous chapter. This first stage will enable us to

deal more easily with the next stage because we have a clear idea of the problem.

Some of the documents consulting during the documentation stage are named bellow:

- TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. Apr 2008. Prashanth Mohan, Venkata N. Padmanabhan and Ramachandran Ramjee. [12]
- Computing Euler angles from a rotation matrix. Gregory G. Slabaugh. [13]
- Using the Rotation Vector Sensor. Android API. [14]
- Dynamics of Tree-Type Robotic Systems, Springer, 2013. Suril Vijaykumar Shah, Subir Kumar Saha, Jayanta Kumar Dutt. [15]

Stage 2 – Training: In this stage, we decided to analyse the platform concerned to the project, Android, studying the features of this platform. Using Android Studio software, that is considered the IDE (Integrated Development Environment) official for the development of Android applications, joined to the knowledge gained during the degree in programming languages like Java, we can understand the theoretical aspect of Android.

The resources used for training in Android are:

- Course Android of the subject Mobile applications. Celeste Campo Vázquez and Carlos García Rubio, 2015. [16]
- Android Programming Course. Salvador Gómez Oliver, November 2011. [17]
- Android API. [14]
- Android 100%. Ramon Invarato Menendez, October 2014. [18]

Stage 3 – Implementation: With the training acquired in the second stage, we proceeded to develop the application. For this use, case and requirements are created and exposed. Furthermore, we created different modules and it will help us to learn how to use the sensors of the Smartphone, as well as the files that the application requires. Knowing its operation, we can proceed to make a model in the testing stage.

Stage 4 – Testing and correction: In stage 4 we made a test, public and private, on the model created in stage 3. Then, we proceeded to solve existing errors or improve the operation of the application**.** The test will be detailed in chapter 5.

Stage 5 – Data analysis with MATLAB: Once the test was carried out and all the problems were solved, the data obtained during the test were analysed with MATLAB software.

Stage 6 – Final application and documentation: During this stage, we finished the application and we proceeded to make a final version of the current document considering the notes taken during de previous stages.

1.4. Resources used

This project has been divided into different stages that are detailed below. It is important to clarify that the stages of the documentation, training and a part of the implementation**.**

During the fulfilment of the project, I have used hardware, software and vehicles. In this section I will explain in general terms, the resources applied and its usefulness (for a more detailed description about the resources, see chapter 6).

Smartphone: different types of Smartphones have been used, testing the application in real time. In all the Smartphones tested, an Android 4.2 (Jelly Bean) software was required or a better one.

Computer: I have used different laptop computers. Its main use enabled the application development and the material in this report. The SO used was Windows and MacOS. As regards to software, I have also used Android Studio, Matlab and Office.

Vehicle: For the road tests I have used different types of vehicles. In order to have a wider sample range, I have used different vehicles in kinds and age.

1.5. Structure of the report

In this section, I will make a general description of the structure of this report.

I**ntroduction and objectives:** A brief introduction to the project is displayed in this first chapter: localization of bumps on the road as well as the motivations to do so. Due to this, I will describe the different objectives to be achieved, the development phases of the project, the resources used and finally, the structure of the document.

State of the Art: In this chapter, I will analyse the technological context of TFG (End of Degree Project). For this reason, the means used will be analysed (devices, technology and roads), as well as the current solutions and the regulatory framework. At the end of this chapter, all the necessary knowledge to understand the solutions explained in previous chapters will be available, recognising the limitations that the technologies used may have.
Analysis and user requirements: In the progress of this chapter I will analyse and design the selected technical solution. I will present the chosen structure, analyse the requirements and I will set forth the use case.

System implementation: Throughout the chapter, I will consider the general aspects of the system, explaining the starting of the application, its development and research. Furthermore, I will account for the theoretical basis that gives rise to implementation by showing flow charts to help in the understanding of the application. I will also explain the internal logistics.

Evaluation and results: In this chapter, I will expose the tests done in order to check that the application meets the requirements previously mentioned.

Project Management: In this chapter, I will show the project management. I will plan and guide the different process that have been followed in the development of this project. I will analyse the different stages, adding a Gantt diagram for a better understanding about the stages of the project. Moreover, the personal costs, material costs and other costs will be defined and detailed.

Conclusions and further improvements: In this chapter, I will give in detail the final conclusions for the making of this project. Besides, future lines of work to improve the application will be exposed.

Appendix A - User Manual: This appendix helps as a brief guide for the user. In it, you can find the different steps to follow in order to use the application.

Appendix B - Glossary: In this chapter, you can find the meaning of the acronyms used throughout the project.

English version: According to the rules applied to the students of Bologna plan (plan 2011), it is necessary to present the following parts in English:

- Complete introduction in English.
- Project summary in English. 5 to 10 sheets. From Chapter 2 to Chapter 6 inclusive.
- Complete conclusions in English.

Bibliography: Explanation of the sources consulted for the completion of the project.

2. STATE OF THE ART (SUMMARY)

Once we have stated the introduction, the next step is to make a brief analysis of the technological context in which this Project is developed. From the definition of the operating system as the one able to interpret what the user wants the terminal to do, we ensure that there are different SO in the market where two of them are mainly highlighted: Android and iOS. Both have different characteristics and although both of them have many possibilities, my choice to develop this application is the Android platform. This platform provides an open code, a

customized interface and a Java language together with the fact that a high percentage of users have a terminal with this kind of operating system.

As we move towards this context and in particular Android, we must state that this operating system is composed by the core of Linux or Kernel which is the software stack layers that manages memory resources, security and drivers; Android Runtime for the execution environment of the application; Libraries/ Bookshops installed by the Smartphone manufacturer; the application Framework or the pattern used for the development or implementation of them until we reach the proper Application, which is the last layer of the software.

Connected to the specific operating system, all the current mobile devices have a great variety of sensors and modules to simplify daily tasks and collect a wide range of data. Among those sensors which to a larger extent affect to this project (bump detection) we emphasise: the accelerometer, whose function is to measure the acceleration with which the telephone moves linearly and the gyroscope, which measures the turn of the device in a diagonal direction. Every time a GPS is on, a calculation process begins following these steps: execution of the internal software able to calculate the location, satellite search, clock synchronisation (UTC), position 3D calculation, and location tracking.

Taken into account those fundamental aspects of one of the essential elements of this project, which is the mobile phone, it is necessary to comment on another key element which is the road conditions.

According to the different studies, the investment in paving and maintenance of roads is minimum and the road conditions get worse over the years. This means negative impacts in the deterioration of the vehicle as well as regarding road safety which increase the accident risk. Contributing to the maximum dealing with vehicle safety, several organisations together with the users started up prevention campaigns to denounce the poor road conditions as in Ponle Freno, launched by Atresmedia group to detect signs and roads in bad conditions. Also, CEA Foundation with Roads or traffic signs in poor conditions; or the European Drivers Association (AEA) with its slogan Tell the Minister!, Tell the Mayor!, Tell the Autonomous Minister! To establish a better communication among drivers and the Government regarding the improvement in infrastructure as they consider this is directly associated with accidents.

Over the last years, taking into account the need and the importance of the pavement surface condition to improve road safety and the number of users that deal with a mobile device, several applications have been created to detect these damages. This is the case of Street Bumps, developed for iOS or *Avisos Madrid* for Android and iOS but which bump identification is done manually (with the inconveniencies it involves). Both applications were tested thanks to the collaboration of the users.

As the potential application for bump detection and the ones already developed need the citizen's cooperation, we will keep in mind the laws that guarantee their privacy and data protection. This is included in the Spanish

Constitution in general (article 18) and more precisely in the Organic Law of Personal Data Protection which determines in its article 1 and I quote it directly "the objective of this law is to guarantee and protect, as regards personal data treatment, public freedoms and the fundamental rights of private individuals, especially their honour and personal and familiar privacy".

I can continue with my project in terms of this regulatory framework, but as we move forward, all the information will be sent through social networks and I should have to consider Law on Telecommunication 9/2014, 9th May to make sure and endorse the secrecy of communications through an encryption system (article 43).

3. ANALYSIS AND USER REQUIREMENTS (SUMMARY)

This section will help to identify the necessities of the project for the user. In order to do so, we will analyse three parts: use cases, user stories and restriction requirements.

Use cases: Our objective is to describe the actions that should be taken place if we want to carry out an action apart from the interaction with the user. Different use cases are distinguished:

- UC-01 Start of the application.
- UC-02 Loading of the setting by the user.
- UC-03 The user selects a new setting.
- UC -04 The system to detect bumps is set up.
- UC -05 Save the setting.
- UC -06 Detect bumps manually, the user can inform about these bumps.
- UC -07 Detects bumps automatically.
- UC -08 Data storage.
- UC -09 Data collection to be analysed by the developer.

User stories: They are an alternative to the functional requirements, written in simple and easy to understand language for the user. It is structured as User Stories with its respective tasks.

- US-01: Detect bumps automatically.
- US-02: Detect bumps manually.
- US-03: Creates information of the recorded bumps.
- US-04 Check the settings.

Restriction requirements: The restriction requirements must take place if we want the application to work as we expect.

4. DESIGN AND DEVELOPMENT OF THE APPLICATION (SUMMARY)

The interest of this application resides in the union of two elements that belong to the daily routine of millions of people, the use of Smartphone and the vehicle, coordinating both to detect potholes on the road surface.

The theoretical concepts of this proposal are based on the study of the article "TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones". [12]

TrafficSense looks for solutions to different problems on the road, such as potholes or traffic events, using a sensor on the Smartphone. In the case of potholes, it will be made using the accelerometer, which detects accelerations on the different Cartesian axes x, y, z (horizontal, vertical, transverse). It will also be necessary to define a reference system for the vehicle, defined by the letters X, Y, Z (front, right and down respectively).

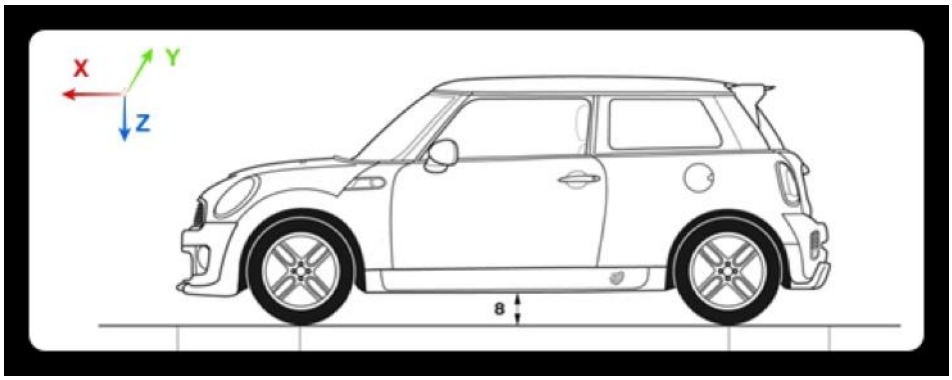


Figure C.2. Cartesian axes car.

In case of unexpected change in the Z-axis, these could be due to potholes or irregularities in the road. So we have our alternative system using a Smartphone sensor: the accelerometer.

After defining the solution, we face a new challenge. The measures on the Z-axis of the car need to have a correspondence with the measures obtained on the Smartphone, so both must be correctly aligned. It is, therefore, necessary to virtually orient the accelerometer so that, when the sensor takes records, it will correspond to the car. The article proposes to do that with a mathematical solution, use the Euler angles using the Z – Y – Z formulation.

The idea of redirecting the accelerometer is based on obtaining samples of the acceleration at certain times that establish the virtual axes of the device $(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}')$ corresponding to the vehicle axis (X, Y, Z). For this reason, we look for patterns in the vehicle causing acceleration during its use in certain axis, and fixing the reorientation.

On the one hand, when the vehicle is at rest, a constant force of 9.8m/s2 is performed in the Z-axis. On the other hand, the acceleration or braking of the vehicle in a straight line will cause an acceleration in the X-axis. Braking is usually sharper than acceleration, so they will be used for taking some data. Y-axis will be deducted by the orthogonally of the reference system.

I will take the data measuring the speed by using the GPS, and if it decreases drastically, we can consider it as a braking and take data of the accelerometer.

Now, I have enough information to calculate the rotation matrix. It will allow me to transform the acceleration vector a_x, a_y, a_z, in the a_X, a_Y, a_Z vector after the reorientation.

I have used the book "Dynamics of Tree-Type Robotic Systems" [15] to learn the theoretical concepts. It addresses about Euler angles, its conjugations, and its form to express it on the matrix. The matrix that we use, called rotation matrix, must be orthogonal and has determinant equal to 1.

The last step is to obtain the values of a'_x, a'_y, a'_z, using the a_X, a_Y, a_Z values and to calculate the rotation matrix.

I have divided the system structure into three parts: view layer, model and controller layer, and operating system layer. This division will help to develop the application.

View layer or user's interface shows the data that it obtains from the model. Users use it to interact with the application.

The model and controller layer: This layer represents the logic of the application. It takes the data and transforms it into significant elements of the system. In this case, the model obtains the data from the accelerometer and the GPS. The controller has a necessary code to respond to the actions that are requested by the app. The model takes some data and the controller uses it, for example, to communicate to the view layer if a pothole has been detected. The elements of this layer are location unit, motion detection unit, reorientation unit, data storage unit and detection of potholes.

Operating system layer, in this case Android, acts as an intermediary between the software and hardware elements. Moreover, it will provide the necessary libraries to make some tasks efficiently.

I will describe that the units belonging to the model and controller layer are connected with the tasks and user history.

- **Location unit:** This unit (connected with US – 03) needs the GPS to set the location of the detected potholes. I created a test application to understand its operating due to the lack of awareness of this sensor in Android system. In this APP, I look for the latitude and longitude values

that allow me to find the location of the potholes. The main class to handle the location services is Location Manager.

- **Motion detection unit:** For this unit, we will use the accelerometer sensor (US – 01). This sensor will turn to be the key for detecting potholes. As GPS, we do not know how to use this sensor on an Android device so we decided to create a basic test application. The main class to develop using accelerometer sensor is *SensorManager*. This class provides access to the different sensors of the device.

- **Reorientation unit:** This unit is related to US – 01. This unit is based on the previous theoretical concepts and lets us reorient the accelerometer. After we had learnt how to use the accelerometer unit, we created a flow diagram to show reorientation unit needs.

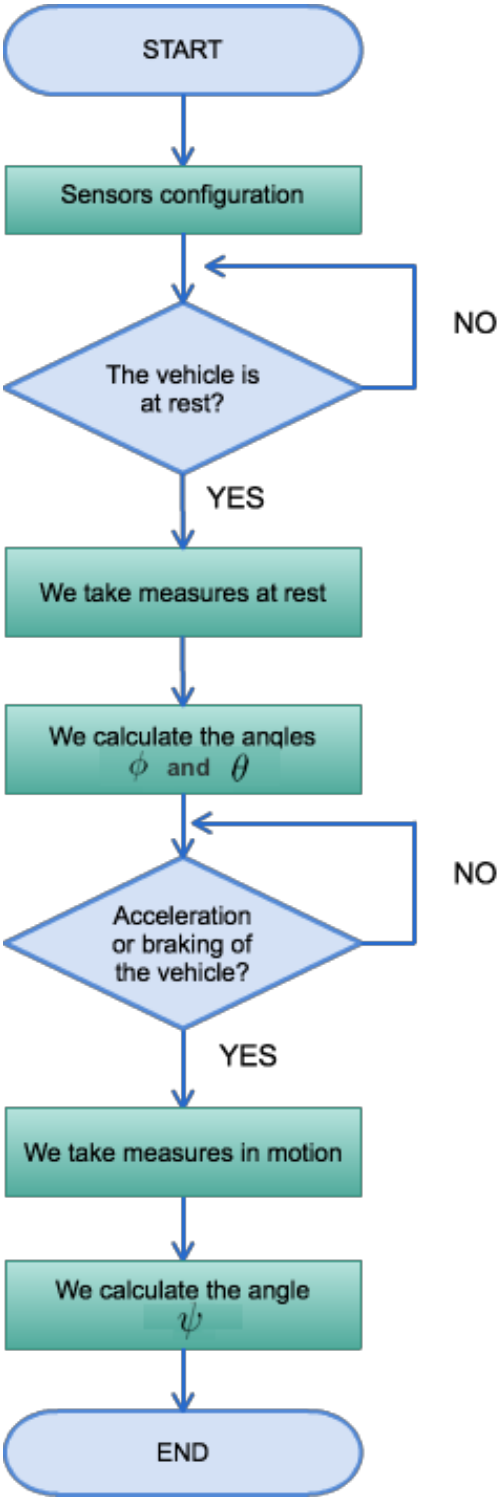


Figure C.3. Flowchart reorientation unit.

This application will take measurements while the vehicle is in rest to calculate the different angles. It will take samples while the vehicle is in motion too. Finally, it will calculate the rotation matrix to reorient the sensor.

- **Data storage unit**: This unit is very useful in the project because it allows us to store data of the detected potholes. For that reason, we will create an application that stores data on the Smartphone memory using data files. *File*, *OutputStreamWriter*, and *FileOutputStream* are the main classes used for this purpose.

To develop all these units, we have created different methods that simplify repetitive tasks. In this way, we make the application more efficient. These classes are *MyApplication* and *Operaciones*.

After developing the different units, we are ready to analyse the different phases to develop the final application.

The first one is called taking and analysing data. This is a first prototype that registers manual events. We use the storage data unit to save accelerometer data and manual events data. With this prototype, we resolve the US – 02, and then, after the analysis, the US – 03.

To start with, we decided to take manual data driving always on the same route. This let us configure the application in a properly way. As we know, the data is stored on data files and we will analyse it using MATLAB software.

After a deep analysis of the data, we can see two levels of variation on the *Z*-axis. These two levels introduce two different concepts of events, potholes and sleeping policemen. Thanks to this analysis, we are ready to establish the operation of the automatic detention on the application. We will create a programme able to detect a_z acceleration variations, and it will decide if there is any pothole event. This new data will be stored on a new data file with the name *auto_potholes*.

The main use of the user's interface is providing a visual and communicative way to interact with the device. That is why it is necessary to create an easy, attractive and intuitive interface. This will help the user to manage the application. This is the appearance of our application.

To represent the different states that rise up when the user performs an action, I use flowcharts, also known as activity diagrams. It is important to say that Android is a system based on *Listener*. These *Listeners* create a parallel flow to the main one when an event happens. I have done different flowcharts:

- **Main flow:** It starts and finishes the application. If this flow is not initiated, the rest will not be launched.
- **"Pothole" and "Sleeping policemen" button flow:** It detects if some of the buttons have been pushed indicating a manual event.
- **"Calibrate" button flow**: detects if the user pushes the button forward to perform a calibration.
- **Automatic detection of potholes flow:** It provides the main operating of the application.

5. TEST AND ANALYSIS RESULTS (SUMMARY)

After the development of the final APP, some private and public tests will be carried out to ensure the application performance.

Private tests: These tests have been done by the developer, making basic and complex tests to verify the proper functioning.

- **Area of private tests:** On the one hand, the developer uses virtual environment (Android Studio) and a physical one (Smartphone) in order to make small tests during the process. On the other hand, the developer makes real tests in the city as well as in low-traffic areas to check reorientation and the performance of manual and automatically bump detection.
- **Basic user's tests:** It is based in the different areas and stories explained in chapter 3, testing those who affect to the functioning of the final application. These tests are useful to demonstrate that all the user's requirements have been fulfilled.

Public tests: These tests required user's cooperation, with a total of 100 volunteers testing this application, filling in a form to have their own point of view.

- **Questionnaire and results:** This questionnaire has been elaborated with the application GOOGLE FORMS, developed by Google to share the information with the volunteers that use the application. The final conclusions are the following:

  - 100% find the application useful.
  - 60% compared to 40% decided to calibrate the application.
  - 80% compared to 20% think it is an easy application.
  - 40% compared to 60% find bumps manually.
  - 20% compared to 80% have already charged a previous calibration.
  - 80% compared to 20% store the calibration

- **Sample analysis:** Once the user's reports are received, an analysis takes place by choosing three random samples to check if there are malfunctions.

6. PROJECT MANAGEMENT (SUMMARY)

The development of the project "Android Application to get data about the state of the roads" has been carried out throughout 10 months combining it with other university duties.

The time estimate dedicated to the project has been done by calculating the hours spent in every application stage with some corrections in the end in order to be close to reality.

Furthermore, there have been meetings during the project elaboration in which the developer and the mentor worked together to make a supervision of the progress of EOD.

Every project requires a budget which includes not only staff costs in terms of working hours, but also material costs and many more that can arise while doing the project. Three engineers were involved in this project; as regards material costs, we considered all the ones related to hardware and software (mobile device, computer, licenses…) and in other expenses we have included a vehicle, fuel, internet tariff, etc.

7. CONCLUSION AND FUTURE IMPROVEMENTS

In this chapter, we will explain in detail the conclusions achieved after the making of this project. Furthermore, we will present the future work lines to improve the application.

7.1. Conclusions

The conclusions of this statement, not only affect to the define objectives, but also to all the knowledge acquire during the development of this project.

After all the test and analysis of the results, we conclude that this project has been a successful one. But as a way to confirm this success, we decided to make a resume through the different objectives we define at first.

On the one hand, this bachelor thesis looked for an accuracy solution that saves money, in comparison with other solutions. We reach this goal using one gadget that is easy to get for millions of people: a Smartphone. The solution must be useful and easy, that is why we chose Android. As we saw in the state of the art, Android is the platform that best fit with our purposes. The protection of user's privacy has been a must during all the development. In addition, we were looking for a solution that detects potholes automatically. We reach this objective developing our own algorithm.

The last objective is reducing the number of accidents. Although it is very difficult to determinate the impact of this APP in the reduction of the number of traffic accidents, this is without any doubt my personal motivation. Trying to find a solution that helps to save hundreds of people life is one of the most amazing challenges. This project may be the future of road condition detection. For sure, that is the most important objective I would like to reach.

7.2. Future improvements

Despite the fulfilment of the objectives while developing the application, some suggested improvements for a more complete performance have arisen. Besides, it has been taken into account the user's opinions from the application in order to achieve an application closer to the user's choice. Some of the future improvements are:

- Show on the screen the number of potholes detected during the use of the application.
- The automation of calibration with no user interaction.
- To implement the sending of data through the communications network allowing users to report the bumps they detected.
- To implement a GPS Navigator in every application or accomplish the fact that this may work at a second level so that users may take advantage of other applications.
- Develop this application in the different Mobile operating systems, with preference for iOS due to its greater market size towards the rest.
- Incentive programme to encourage users to use this application.

BIBLIOGRAFÍA

[1] ”IBM Simon”, *Wikipedia*. [En línea]. Disponible en: https://es.wikipedia.org/wiki/IBM_Simon

[2] Dirección general de carreteras e infraestructuras, **“**Estudio de la Intensidad Media Diaria (IMD) de circulación 2015”, Consejería de transportes, viviendas e infraestructuras, C. de Madrid, España, 2015. [En línea]. Disponible en: http://www.madrid.org/cs/Satellite?c=CM_InfPractica_FA&cid=1354426677832&language=es&pagename=ComunidadMadrid%2FEstructura

[3] J.Jiménez Gálvez, **“**El deterioro de las carreteras, cifrado en 6.600 millones de euros”, *El País*, 05-05-2016. [En línea]. Disponible en: http://politica.elpais.com/politica/2016/05/05/actualidad/1462454369_078076.html

[4] A. Martín, “El 10% de los accidentes se deben al mal estado de las carreteras”, *20 Minutos*, 12-09-2006. [En línea]. Disponible en: http://www.20minutos.es/noticia/151094/0/accidentes/estado/carreteras/

[5] E. Cano, “Falta mantenimiento en las carreteras españolas”, *ABC*, 04-11-2015. [En línea]. Disponible en: http://www.abc.es/motor/reportajes/abci-aumenta-riesgo-accidente-falta-mantenimiento-carreteras-espanolas-201511040216_noticia.html

[6] J. Jiménez Gálvez, “El riesgo mortal de volver a las carreteras”, *El País*, 11-07-2015. [En línea]. Disponible en: http://politica.elpais.com/politica/2015/07/11/actualidad/1436571260_237661.html

[7] A. Westlake, “Land Rover develops tech to warn you and the city about photoles”, *Slash Gear*, 11-06-2015. [En línea]. Disponible en: http://www.slashgear.com/land-rover-develops-tech-to-warn-you-and-the-city-about-potholes-11387923/

[8] D. Murias, “La suspensión Bose era tan innovadora que permitía al coche saltar por encima del obstáculo”, *Motor Pasión*, 07-02-2016. [En línea]. Disponible en: http://www.motorpasion.com/tecnologia/la-suspension-bose-era-tan-innovadora-que-permitia-al-coche-saltar-por-encima-de-un-obstaculo

[9] M. Dorta, “*Ponle Freno* busca carreteras en mal estado y señales defectuosas”, *Antena 3*, 09-05-2013. [En línea]. Disponible en: http://www.antena3.com/ponlefreno/ponle-freno-vuelve-buscar-senales-peligrosas_2011020400060.html

[10] Comisariado europeo del automóvil, “Carreteras en mal estado o señales defectuosas”, *Fundación CEA*. [En línea]. Disponible en: **[**http://www.fundacioncea.es/denuncia-carretera-mal-estado.asp**]**

[11] Automovilistas Europeos Asociados, “Denuncias sobre el estado de las carreteras”, *Automovilistas Europeos Asociados (AEA)*. [En línea]. Disponible en: http://aeaclub.org/denuncias-seguridad-vial

[12] P. Mohan, V.N. Padmanabhan and R. Ramjee. “TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones”, *Technical Report MSR-TR-2008-59*, Bangalore, India, 2008. [En línea]. Disponible en: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2008-59.pdf

[13] G.G. Slabaugh "Computing Euler angles from a rotation matrix", City University London, Londres, Reino Unido, 2015. [En línea]. Disponible en: http://www.staff.city.ac.uk/~sbbh653/publications/euler.pdf
[14] "Developers Android", *Google*. [En línea]. Disponible en: https://developer.android.com/index.html
[15] S. Suril Vijaykuma, S. Kumar Saha, J. Kumar Dutt, "Capítulo 3: Euler-Angle-Joints", en *Dynamics of Tree-Type Robotic Systems*, 2013 ed. Springer, 2013.
[16] C. Campo Vázquez y C. García Rubio, Curso de Android de la asignatura Aplicaciones móviles, Universidad Carlos III de Madrid, Leganés, 2015.
[17] S. Gómez Oliver, *Curso Programación Android*, 3th ed. Publicación libre, 2015. [En línea]. Disponible en: http://www.sgoliver.net/blog/curso-de-programacion-android/curso-en-pdf/
[18] R. Invarato, "Libro Android 100%", 1st ed. Publicación Libre, 2016. [En línea] Disponible en: http://jarroba.com/libro-android-100-gratis/
[19] "Sistema operativo móvil", *Wikipedia.* [En línea]. Disponible en: https://es.wikipedia.org/wiki/Sistema_operativo_móvil
[20] A. Mocholí, "Crear apps móviles: Diferencias entre Android e iOS", Ana Mocholí, *Yeeply*, 10-11-2015. [En línea]. Disponible en: https://www.yeeply.com/blog/crear-apps-moviles-diferencias-android-e-ios/
[21] L. Cancela García y S. Ostos Lobo, "Arquitectura Android", *Tutorial Android para Software de Comunicaciones*, Universidad Carlos III, Leganés, España. [En línea]. Disponible en: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android
[22] Y. Aparicio, "Barómetro, giroscopio y otros sensores de tu Smartphone", *PandaAncha*, 29-10-2015. [En línea]. Disponible en: http://www.pandaancha.mx/noticias/barometro-giroscopio-sensores-smartphone.html
[23] L. El Asri, "Así funcionan las tripas de tu móvil: el acelerómetro, un sensor que te puede salvar la vida", *eldiario.es*, 26/06/2014. [En línea]. Disponible en: http://www.eldiario.es/hojaderouter/tecnologia/acelerometro-funciones-giroscopio-GPS-interior-magnetometro-sensor-sensor_de_humedad-sensor_de_temperatura-telefono_movil_0_275772515.html
[24] I. Lerguera Valencia, "¿Qué sensores incorpora tu Smartphone?", *TuTecnoMundo*, sep-2015. [En línea]. Disponible en: https://www.tutecnomundo.com/que-sensores-incorpora-tu-smartphone/
[25] "Giroscopio", *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Giróscopo]
[26] D. Pérez, "Todo sobre el GPS en Android: como funciona y como desactivarlo", *El Android Libre*, 28-10-2015. [En línea]. Disponible en: http://www.elandroidelibre.com/2015/10/todo-sobre-el-gps-en-android-como-funciona-y-como-desactivarlo.html
[27] BQ, "GPS en el Aquaris 5", *Mi Bq y yo*, 5-09-2013. [En línea]. Disponible en: http://www.mibqyyo.com/articulos/2013/09/05/gps-en-el-aquaris-5/#/vanilla/discussion/embed/?vanilla_discussion_id=0
[28] M. Rodrigo y S. Rubio, "Estudio sobre las Necesidades de Inversión en Conservación", *Asociación Española de la Carretera*, 2016. [En línea].

Disponible en: http://www.aecarretera.com/estudios/estudios-tecnicos/estudios/2696-estudio-sobre-necesidades-de-inversion-en-conservacion2016
[29] ASEFMA, "ASEFMA recuerda que el estado de conservación de las carreteras afecta directamente al bolsillo de los ciudadanos", *Asociación Española de Fabricantes de Mezclas Asfálticas*, 3-06-2013. [En línea]. Disponible en: http://www.asefma.es/asefma-recuerda-que-el-estado-de-conservacion-de-las-carreteras-afecta-directamente-al-bolsillo-de-los-ciudadanos/
[30] Street Bump, "Where is Street Bump being used?", *Street Bump*. [En línea]. Disponible en: http://www.streetbump.org
[31] Área de Gobierno de Economía, Hacienda y Administración Pública, "Avisos Madrid", *Ayuntamiento de Madrid*. [En línea]. Disponible en: http://www.madrid.es/portales/munimadrid/es/Inicio/El-Ayuntamiento/Atencion-a-la-ciudadania/Aplicacion-movil/Aplicacion-movil-Avisos-Madrid-?vgnextfmt=default&vgnextoid=c67bd881111f7410VgnVCM1000000b205a0aRCRD&vgnextchannel=b9254bbed88f7410VgnVCM1000000b205a0aRCRD
[32] Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal. BOE 298 , 14 diciembre 1999. [En línea]. Disponible en: https://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf
[33] Ley 9/2014, de 9 de mayo, de Telecomunicaciones, BOE 114, 10 mayo 2014. [En línea]. Disponible en: https://www.boe.es/boe/dias/2014/05/10/pdfs/BOE-A-2014-4950.pdf
[34] "Historias de usuario", *Wikipedia*. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Historias_de_usuario
[35] *IEEE Recommended Practice for Software Requirements Specifications*, 830-1998. [En línea]. Disponible en: https://standards.ieee.org/findstds/standard/830-1998.html
[36] "Matlab", *MathWorks*. [En línea]. Disponible en: http://es.mathworks.com/products/matlab/
[37] "Diccionario de la lengua española", *Real Academia Española*. [En línea], Disponible en: http://www.rae.es
[38] "Google Forms", *Google*. [En línea]. Disponible en: https://www.google.es/intl/es/forms/about/
[39] A. González Caballero, "Aplicación Android para obtener información de tráfico en carreteras", Trabajo de Fin de Grado, Dpto. Ingeniería Telemática, Universidad Carlos III, Leganés, España, 2016.